

Overview

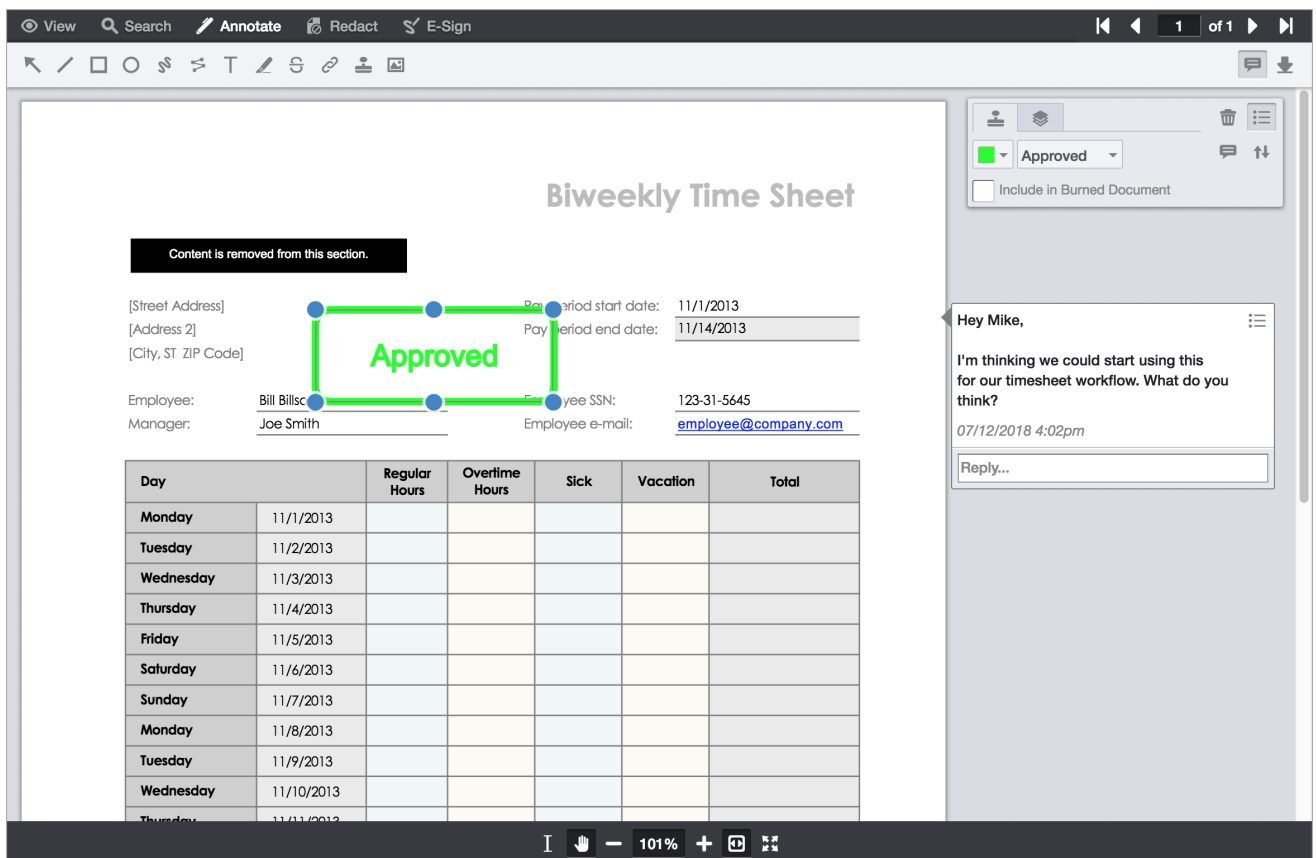


Introduction

PrizmDoc Viewer allows you to add powerful **document viewing** and **document conversion** functionality to your web applications.

Document Viewing for the Browser

PrizmDoc Viewer includes an advanced HTML Viewer control which allows your users to view, search, annotate, redact, print, and download documents in [many different file formats](#), right in their browser. They don't need to leave your application or install any custom software:



Seamless Integration

Our viewer is designed for seamless integration with your web application. Key features, like search and redaction, can be easily turned on or off depending on your application's needs (see the [uiOptions Object](#)). If you don't like the out-of-the-box UI layout or style, you can [completely change it](#). And since the viewer has an extensive [JavaScript API](#), your application can programmatically control and respond to the viewer.

Powerful Features

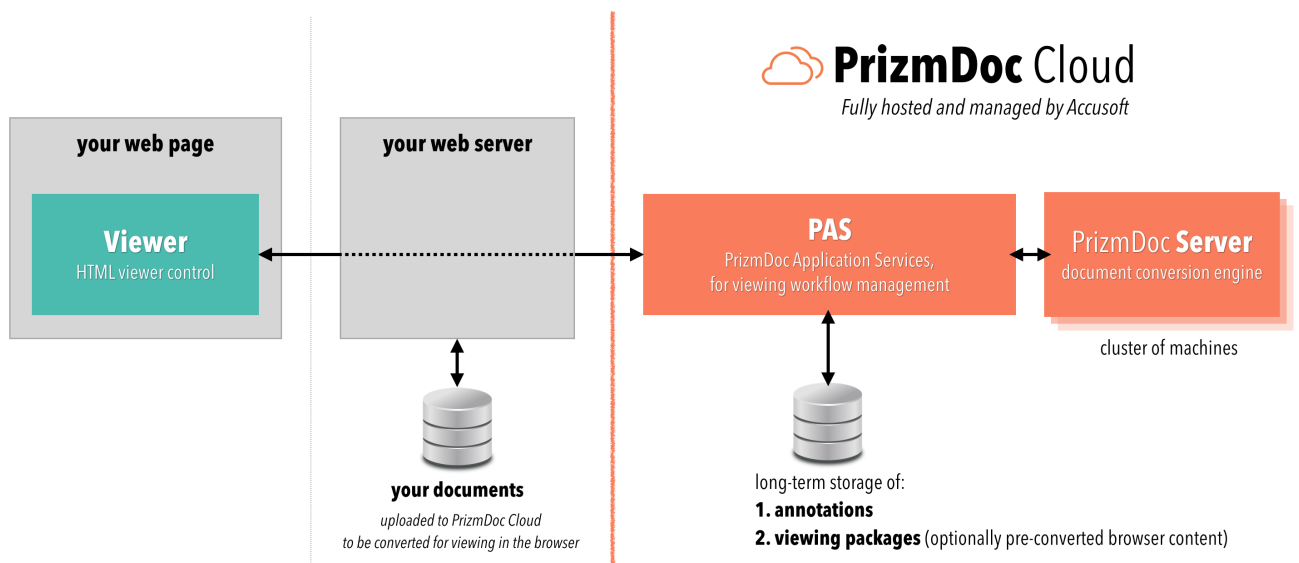
PrizmDoc Viewer offers powerful viewing features, including:

- The option to use [Microsoft Office](#) for high-fidelity rendering of Word, Excel, and PowerPoint files (included automatically with PrizmDoc Cloud)
- Support for [viewing and searching of large documents](#) with thousands of pages
- The ability to [review changes between two different Microsoft Word files](#)
- The option to [pre-convert document content in advance](#) for even faster viewing on the web

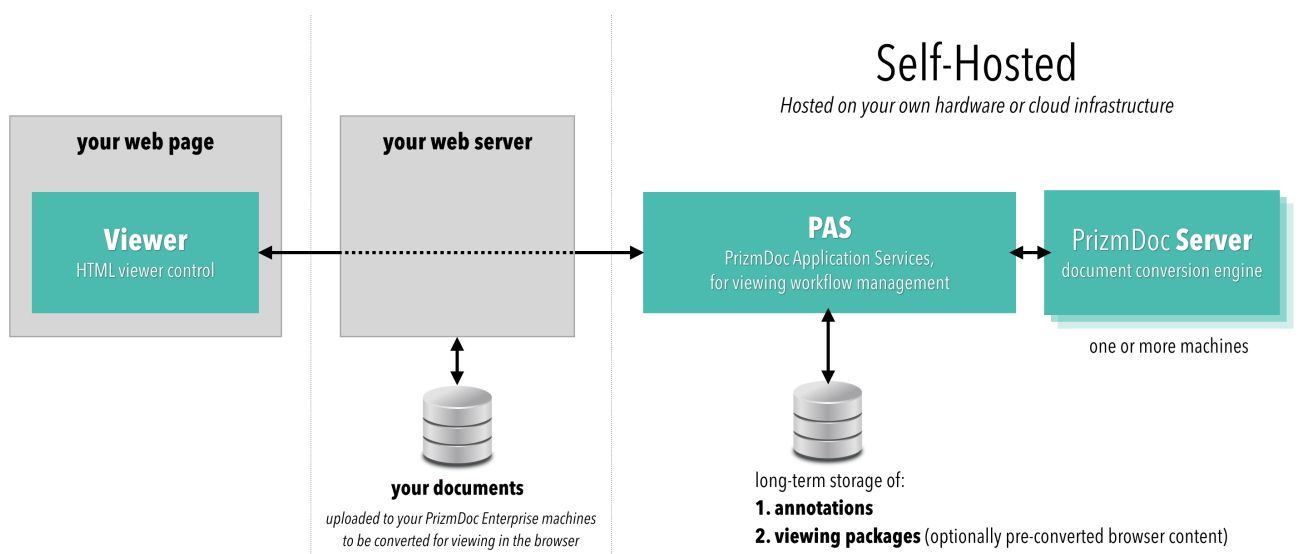
Overall Architecture

Viewing functionality is powered by 1) the HTML Viewer control and 2) powerful REST APIs and server-side software.

The easiest way to get started is with [PrizmDoc Cloud](#), where we fully host and manage the server-side pieces of PrizmDoc Viewer for you:



Of course, if you need to, you can self-host everything:



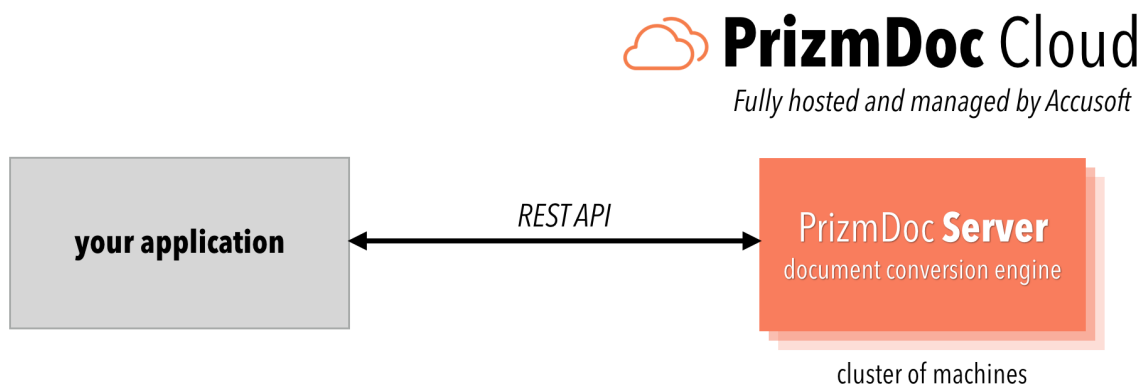
Document Conversion

You can use PrizmDoc Viewer's [Content Conversion Service REST API](#) to easily:

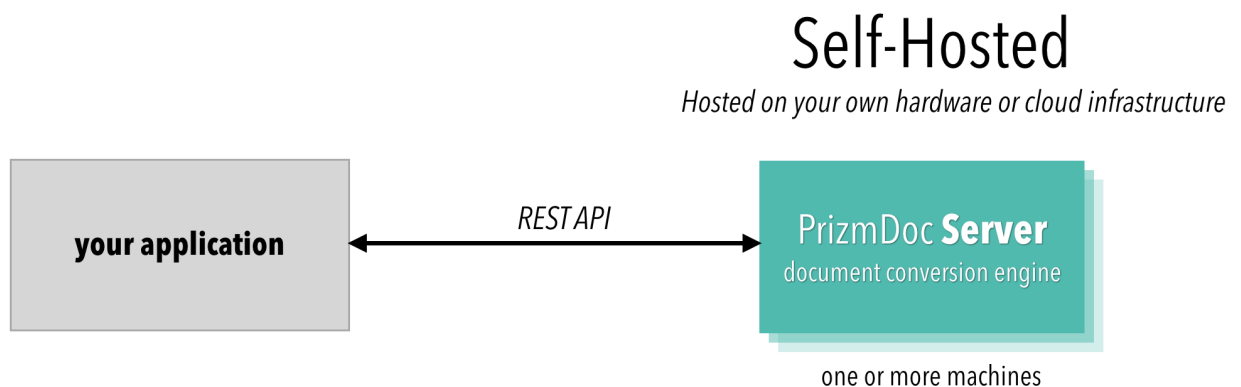
- Convert from Office, PDF, Email, HTML, TIFF, PNG, JPEG, CAD, and [many other kinds of documents](#) to PDF, TIFF, PNG, JPEG, or SVG
- Use OCR to convert a non-searchable PDF or TIFF into a new, visually-identical PDF that supports full-text search
- Split and merge pages from multiple documents
- Apply headers, footers, or watermarks to a document

As with viewing, the conversion REST APIs offer the option to use [Microsoft Office](#) for high-fidelity conversion of Word, Excel, and PowerPoint files (included automatically with PrizmDoc Cloud).

The easiest way to get started is with [PrizmDoc Cloud](#), where we fully host and manage a powerful PrizmDoc Server cluster for you:



Of course, if you need to, you can self-host the server:



Supported File Formats

Supported File Formats

This section represents a reference for every document and image file format supported by PrizmDoc.

PrizmDoc detects most of the formats automatically, except the formats with poor signature or no signature at all. Such formats are detected by the file extension.

PrizmDoc Viewer disables JavaScript execution for any HTML file conversion.

Document Formats

Format	File Extension	Supported by Microsoft Office renderer	Auto Detection
Adobe Portable Document format	*.pdf	No	Yes
Microsoft Word format	*.doc, *.dot	Yes	Yes
Microsoft Word Open XML format	*.docx, *.docm, *.dotx, *.dotm	Yes	Yes
Rich Text format	*.rtf	Yes	Yes
Microsoft Excel format	*.xls, *.xlt	Yes	Yes
Microsoft Excel Open XML format	*.xlsx, *.xlsm, *.xltx, *.xltm	Yes	Yes
Microsoft PowerPoint format	*.ppt, *.pot, *.pps	Yes	Yes
Microsoft PowerPoint Open XML format	*.pptx, *.pptm, *.potx, *.potm, *.ppsx, *.ppsm	Yes	Yes
OpenDocument Text format	*.odt, *.ott, *.fodt	No	Yes
OpenDocument Spreadsheet format	*.ods, *.ots, *.fods	No	Yes
OpenDocument Presentation format	*.odp, *.otp, *.fodp	No	Yes
OpenDocument Math Formula format	*.odf	No	Yes
OpenDocument Drawing format	*.odg, *.otg, *.fodg	No	Yes

CAD Formats

Format	File Extension	Auto Detection
AutoDesk AutoCAD format (version 2.5 through 2014)	*.dwg, *.dxf	Yes
AutoDesk Design Web format	*.dwf	Yes
MicroStation Drawing format (V7 and V8)	*.dgn	Yes

Web Formats

PrizmDoc Viewer disables JavaScript execution for any HTML file conversion.

Format	File Extension	Auto Detection
HyperText Markup Language format	*.html, *.htm	Yes

Format	File Extension	Auto Detection
Extensible HyperText Markup Language format	*.xhtml, *.xhtm	Yes

Most of the HTML files are auto-detected by searching for specific tags. Although html tags usually denote a web page, they neither guarantee that the file is a web page, nor they are required for rendering a file as a web page. Such incompliant files might not be recognized as HTML, but will still be rendered as HTML in case if they are identified as such by the file extension.

Email Formats

Format	File Extension	Auto Detection
Microsoft Outlook format	*.msg	Yes
Outlook Express Email format	*.eml	No

EML files are not auto-detected, but identified by file extension because EML is a plain text in MIME format which does not allow reliable auto-detection. Currently PrizmDoc supports rendering of the following email headers: From, Subject, To, CC, BCC, Sent, and Attached.

Text Formats

Format	File Extension	Supported by Microsoft Office renderer	Auto Detection
Text format	*.txt	No	No
Comma-Separated Values format	*.csv	Yes	No

Text and CSV files are not auto-detected, but identified by file extension because both are plain text formats with no file signature. These formats however assume different rendering style. While Text is rendered plain, the CSV is rendered like a spreadsheet. We distinguish them by file extension.

Medical Image Formats

Format	Compression	File Extension	Auto Detection
Digital Imaging & Communication in Medicine format	Uncompressed, JPEG, RLE	*.dcm, *.dicom, *.dcim, *.dicm	Yes

Currently PrizmDoc auto detects DICOM Specification Part 10 compliant files only. Auto-detection of the legacy DICOM files (without DICOM File Meta Information) is not supported, therefore the legacy DICOM files are detected by the file extension. Currently PrizmDoc does not support conversion of DICOM format to SVG.

Image Formats

Format	Compression	File Extension	Auto Detection
Tagged Image File Format	Uncompressed, PackBits, Huffman, CCITT G3, CCITT G4, CCITT G32D, JPEG, Deflate, LZW	*.tif, *.tiff	Yes
JPEG File Interchange Format	JPEG	*.jpg, *.jpeg	Yes
JPEG 2000 File Format and Code Stream	JPEG 2000	*.jp2, *.jpc	Yes
Graphics Interchange Format	LZW	*.gif	Yes
Portable Network Graphics format	Deflate	*.png	Yes
Adobe Photoshop format	Uncompressed, Deflate, RLE	*.psd, *.psb	Yes
Microsoft Windows Bitmap format	Uncompressed, RLE	*.bmp, *.dib	Yes
Macintosh Metafile format	Uncompressed, RLE, JPEG	*.pct, *.pic, *.pict	Yes
Windows Metafile format (see note below)	Uncompressed, RLE	*.wmf	Yes
Enhanced Metafile format (see note below)	Uncompressed, RLE	*.emf	Yes
ZSoft Paintbrush PCX format	Uncompressed, RLE	*.pcx	Yes
ZSoft Paintbrush DCX format	Uncompressed, RLE	*.dcx	Yes
Sun Raster Data format	Uncompressed, RLE	*.ras	Yes
Kodak Photo CD format	Uncompressed, Huffman	*.pcd	Yes
Truevision Targa format	Uncompressed, RLE	*.tga, *.tpic	Yes
Continuous Acquisition and Life-cycle Support format	CCITT G4	*.cal, *.cals	Yes
Icon Resource format	Uncompressed, RLE	*.ico	Yes
Windows Cursor format	Uncompressed, RLE	*.cur	Yes
NCR Image format	Uncompressed, CCITT G4	*.ncr	Yes
X Window Dump format	Uncompressed	*.xwd	Yes
Silicon Graphics Image format	Uncompressed, RLE	*.sgi	Yes
Wireless Bitmap format	Uncompressed	*.wbmp	Yes

Format	Compression	File Extension	Auto Detection
Scitex Color Tone format	Uncompressed	*.sct	Yes
WordPerfect Graphics Metafile format	RLE	*.wpg	Yes
X Bitmap format	Uncompressed	*.xbm	Yes
Portable Bitmap format	Uncompressed	*.pbm	Yes
Portable Graymap format	Uncompressed	*.pgm	Yes
Portable Pixmap format	Uncompressed	*.ppm	Yes
Xerox 9700 Graphic format	Uncompressed	*.img	Yes
Dr. Halo format (see note below)	RLE	*.cut	No

- PrizmDoc Server running on Windows supports vector and raster content in Windows Metafile (WMF) and Enhanced Metafile (EMF) formats. On Linux platforms, only WMF and EMF files with a single raster image are supported; all others are rejected.
- Dr. Halo (CUT) format cannot be reliably auto-detected due to its poor file signature, therefore it is identified by file extension.
- PrizmDoc does not support conversion of CAL/CALS, CUR, DCX, IMG, PCT/PIC/PICT, PSD/PSB, RAS, TGA/TPIC and XWD formats to SVG.

Viewer Requirements

Supported Browsers

CAUTION: PrizmDoc Viewer uses webfonts. We highly recommend that you do not disable the webfonts in your browser. If you disable webfonts (via browser or ad blocker settings), PrizmDoc Viewer will fall back to a compatibility mode which will result in much slower document rendering and scrolling (a warning will be sent to the browser console).

We support the current version and one version back for the following browsers:

iOS

- Safari
- Google Chrome

Android

- Android Browser
- Google Chrome
- Mozilla Firefox

Windows

- Internet Explorer (v11 only)

NOTE: Support for Internet Explorer was deprecated with PrizmDoc Viewer v13.14.

- Microsoft Edge

- Google Chrome
- Mozilla Firefox

Mac OS

- Safari
- Google Chrome
- Mozilla Firefox

Third-Party Dependencies

- jQuery 3.6.0
- Underscore 1.13.1
- (Full Viewer) jQuery.Hotkeys 0.8

PrizmDoc Cells Overview

Introduction

PrizmDoc Viewer's rendering of Excel files is similar to viewing a spreadsheet in print preview. In many cases, this view will suffice but it can be restrictive if you need to view values as well as formulas in a spreadsheet. If you need a more detailed viewing experience of Excel files, we have built an advanced spreadsheet viewer called PrizmDoc Cells to give PrizmDoc Viewer users the ability to review Excel files exactly as they would appear in the native application.

You can analyze formulas, view charts and graphs, view multiple spreadsheets in a single workbook, and navigate without pagination. Spreadsheet content and cells can be searched using the search bar. Search term "hits" are highlighted in real-time as you type in the first character. With PrizmDoc Cells, you are able to view formulas as well as formula results, search for key data and words, and not be restricted to viewing dynamic data as a static image.

PrizmDoc Cells is offered as an option to PrizmDoc Viewer (both Self-Hosted and Cloud) and utilizes the PrizmDoc Content Conversion Service to produce the best fidelity possible when rendering charts and graphs embedded in a spreadsheet.

Integration with PrizmDoc Viewer

For more information on obtaining and integrating PrizmDoc Cells into a PrizmDoc Viewer deployment, view the PrizmDoc Cells integration instructions with links to sample code [here](#).

Evaluation and Deployment Licenses

For your convenience, PrizmDoc Cells automatically runs in evaluation mode without a license. When you are ready to deploy to production, a license will be required to run PrizmDoc Cells in your production environment. Please contact info@accusoft.com to obtain a production license.

Accusoft Support

Introduction

If you have a question, or are experiencing an issue, check our [Troubleshooting section](#) and our online [FAQs](#). Still have a question? Send us a technical support request from our [website](#).

The Accusoft Support Team can help with problems or questions you may have about the following:

- Installation
- Licensing
- Sample code
- Error handling
- General questions or help on how to use PrizmDoc Viewer

For information about Accusoft Support Plans, see [Support Plans](#) or call Accusoft at 813-875-7575.

Also, check out the following additional resources on our [website](#):

- [Online Demos](#)
- [Code Samples](#)

Glossary

This section contains the following information for PrizmDoc Viewer:

- [Definitions](#)
- [New Terms](#)

Definitions

Introduction

The following table lists common terminology for components and important concepts of PrizmDoc Viewer and brief definitions of each, along with alternate names or abbreviations, and links to additional documentation.

Term	Also Known as...	Definition
Application Services	-	The component of PrizmDoc Viewer that provides application-level logic between PrizmDoc Server and the Viewer or the customer's web-tier. See PrizmDoc Application Services (PAS) .
AutoRedaction Service	ARS	A component of PrizmDoc Server that creates markup XML for redactions on a document. See Performing Auto-Redaction .
Central Configuration	Central Config	The configuration file for PrizmDoc Server. See Central Configuration .
Cells	-	PrizmDoc Cells is our spreadsheet viewer that gives PrizmDoc Viewer users the ability to review Excel files exactly as they would appear in the native application.
Cloud Entry Point	CEP	The Load Balancer endpoint for requests to a cluster of PrizmDoc Server instances, to be routed to an appropriate instance.
Cluster Mode	-	A mode in which multiple instances of PrizmDoc Server execute simultaneously in a cluster, with individual Server Entry Points and a Cluster Entry Point that could route to any of them. See PrizmDoc Viewer Cluster Mode .

Term	Also Known as...	Definition
Content Conversion Service	CCS	A component of PrizmDoc Server that provides document format conversion. See Convert Content with CCS .
Email Conversion Service	ECS	A component of PrizmDoc Server that converts files from EML and MSG formats.
Email Processing Service	EPS	A component of PrizmDoc Server that extracts content from EML and MSG files.
Error Reporting Service	ERS	A component of PrizmDoc Server that logs errors originating from other PrizmDoc Server components. See Error Reporting .
Format Detection Service	FDS	A component of PrizmDoc Server that identifies the format of a source document.
HTML Conversion Service	HTMLCS	A component of PrizmDoc Server that converts files from HTML formats.
Imaging Services	PCCIS	A component of PrizmDoc Server that handles the creation and management of viewing sessions.
Load Balancer	PLB	A component of PrizmDoc Server that routes requests to the correct component and balances loads across PrizmDoc Server instances in Cluster Mode.
Office Conversion Service	OCS	A component of PrizmDoc Server that converts files from Office and text formats.
PDF Conversion Service	PDFCS, Imaging Conversion Service	A component of PrizmDoc Server that converts files from PDF format.
PDF Processing Service	PDFPS	A component of PrizmDoc Server that handles text extraction and markup burning for PDF documents.
Prizm License Utility	PLU	A component of PrizmDoc Server that handles licensing of the product on installation. See Licensing .
PrizmDoc Viewer	-	The full product composed the PrizmDoc Server, PrizmDoc Application Services (PAS), and the Viewer.
PrizmDoc Cloud	-	A PrizmDoc Server hosted by Accusoft. See PrizmDoc Server (PrizmDoc Cloud) .
PrizmDoc Self-Hosted	-	PrizmDoc Viewer hosted on a customer's server. See Server Hosting Options .
PrizmDoc	-	A back-end component of PrizmDoc Viewer that performs conversions and other

Term	Also Known as...	Definition
Server		manipulations of source files.
Raster Conversion Service	RCS	A component of PrizmDoc Server that converts files from raster formats.
Redaction Service	-	A component of PrizmDoc Server that handles the markup burning workflow.
Server Entry Point	SEP	The Load Balancer endpoint for requests to a specific PrizmDoc Server instance.
Vector Conversion Service	VCS	A component of PrizmDoc Server that converts files from vector formats.
Viewer	-	The front-end component of PrizmDoc Viewer that allows the uploading and display of documents through a browser. See Viewer.
Viewing Package	-	A cache of web-compatible content for a document immediately available for use in a Viewing Session.
Viewing Session	-	A resource that provides web-compatible content for an uploaded document.
Watchdog	-	A component of PrizmDoc Server that launches and monitors the health of other PrizmDoc Server components.
Web Tier	-	A customer's web tier application that interfaces with PrizmDoc Viewer.
Work File Service	WFS	A component of PrizmDoc Server that handles the uploading and storage of source documents.

New Terms

Introduction

The product names have been updated to the following:

- [PrizmDoc Viewer](#)
- [PrizmDoc Cloud](#)
- [PrizmDoc Self-Hosted](#)
- [PAS](#)
- [PrizmDoc Server](#)
- [Cloud Authentication](#)
- [The Viewer](#)

PrizmDoc Viewer

Formerly referred to as:

- PrizmDoc

PrizmDoc Cloud

Formerly referred to as:

- PrizmDoc Cloud-Hosted
- PrizmDoc Accusoft Cloud-Hosted
- Accusoft Cloud-Hosted Services
- Accusoft Cloud Services (ACS)
- Accusoft-Hosted Services

PrizmDoc Self-Hosted

Formerly referred to as:

- PrizmDoc Enterprise

PAS

Formerly referred to as:

- PrizmDoc Application Services
- Prizm Application Services

PrizmDoc Server

NOTE: When using the PrizmDoc Server APIs, you will see `PCCIS` in the endpoint URL which is shorthand for PrizmDoc Server.

Formerly referred to as:

- PCC Backend Services
- PCC Imaging Services (PCCIS)
- PCC Services
- Prizm Backend Services
- Prizm Imaging Services
- Prizm Platform Services (PPS)
- Prizm Services

Cloud Authentication

Formerly referred to as:

- PrizmDoc Cloud API

The Viewer

Formerly referred to as:

- PrizmDoc Viewer
- Client Viewer
- Document Viewer
- HTML5 Application

- HTML5 Responsive Viewer
- HTML5 Viewer
- PCC Viewer
- Prizm Responsive Viewer
- The Viewing Client
- Viewer Application
- Viewer UI

Legal

This section contains legal information for PrizmDoc Viewer:

- [Copyright Information](#)
- [Software License Agreement](#)
- [Third-Party Attributions](#)

Copyright Information

©2008-2021 Accusoft Corporation. All rights reserved.

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Accusoft® Corporation.

This manual and the software described in it are both products of the United States of America.

Accusoft Corporation
4001 North Riverside Drive
Tampa, FL 33603
Sales: 813-875-7575
info@accusoft.com
www.accusoft.com

Accusoft Trademarks

Visit our [website](#) for a complete list of trademarks (™) and registered marks (®) of Accusoft Corporation.

Accusoft Corporation and/or its agents use these marks and brand names in connection with its goods and/or services, in the United States and other countries.

All other product and brand names are the property of their respective owners.

Accusoft Patents

PrizmDoc Viewer utilizes technology owned by Accusoft Corporation that is protected by U.S. Patents 9,860,194; 9,886,426 and U.S. Patents Pending.

Software License Agreement

ACCUSOFT CORPORATION

PRIZMDOC VIEWER 'SHRINK-WRAP' LICENSE AGREEMENT

PLEASE READ THIS LICENSE AGREEMENT ("AGREEMENT") WHICH GOVERNS YOUR RIGHT TO USE OF PRIZMDOC VIEWER ("PROGRAM"). YOU MUST ACCEPT THESE TERMS BEFORE YOU ARE PERMITTED TO INSTALL THE PROGRAM. YOU EXPRESSLY AGREE THAT YOU HAVE THE AUTHORITY TO CONTRACTUALLY BIND THE ORGANIZATION OR ENTITY YOU REPRESENT ("LICENSEE") TO BE BOUND BY THESE TERMS.

BY DOWNLOADING THE PROGRAM FROM Accusoft AND THEN CONTINUING WITH THE INSTALLATION AND USE OF PROGRAM, YOU AND LICENSEE AGREE TO BE BOUND BY THIS AGREEMENT. AND YOU EXPLICITLY CONFIRM THAT YOU ARE ACCEPTING OUR [PRIVACY POLICY](#). IF YOU HAVE ANY QUESTIONS ABOUT THAT POLICY, PLEASE EMAIL privacy@accusoft.com.

1. Background. Accusoft Corporation, a Florida corporation, ("Accusoft") is the owner of all right, title, and interest in the software system known as PrizmDoc Viewer ("Program") and consisting of an installed front-end component plus either an installed back-end component hosted by LICENSEE or back-end services known as PrizmDoc Cloud ("Service") hosted by Accusoft. LICENSEE desires to receive and use a copy of Program under the terms and conditions stated herein, for the purpose of evaluating the Program under an Evaluation Mode Limited License (Paragraph 3) or for a commercial purpose under a Commercial License (Paragraph 4).

2. Evaluation Mode and Licensed Mode

Default Installation - Evaluation Mode (or Trial Licensing)

The Program installation installs Program in Evaluation Mode. This allows you to test many Program features and functions. Images may be displayed with a watermark on them and occasionally dialogs may be posted reminding you that Program is in evaluation mode. Printed, exported and e-mailed images may also display a watermark. This Evaluation Mode license may expire after 30 days at which time Program may cease operating. If your evaluation is not complete at that time, please contact www.accusoft.com or sales@accusoft.com to see if your Evaluation Mode license time can be extended.

Changing from Evaluation Mode to Licensed Mode

A Commercial License may be purchased at www.accusoft.com or through sales@accusoft.com and then LICENSEE's rights as to the number of installations and scope and term of use are governed solely by the purchased license and **LICENSEE is required to purchase the appropriate license PRIOR TO SUCH INSTALLATIONS.**

Accusoft software applications, including Program, are limited to use on a single computer. No runtimes or copies may be installed or distributed unless that installation or distribution is granted by a direct license from Accusoft. These 'license agreements' provide the terms and limits of number of copies and usage.

All prospective customers have every opportunity to evaluate Accusoft's products including Program prior to purchasing. Accusoft fully supports and warrants its code and its pricing of Program reflects those support and warranty costs.

3. Evaluation Mode Limited License. In Evaluation Mode, Accusoft grants to LICENSEE only a limited, non-transferable, non-exclusive and non-assignable license to evaluate the Program on a single computer for a thirty (30) day period beginning on the date of download of the Program and as may be subsequently extended by Accusoft on LICENSEE's request ("Term"), for the sole purpose of evaluating the Program (the "Purpose"), and not for any commercial usage. For clarity, LICENSEE may only install and use the Program on a single computer, and may only use it in an internal testing or proof-of-concept environment. **LICENSEE IS NOT PERMITTED TO INSTALL AND USE THE PROGRAM IN A PRODUCTION ENVIRONMENT.** Either party may terminate this Agreement for convenience prior to the end of the Term on one day's written notice (email notice is acceptable) to the other party. LICENSEE shall have no right to, and shall not assign this Agreement whether by transfer, assignment, merger or otherwise.

4. Commercial License. A license may be purchased at www.accusoft.com or through sales@accusoft.com. If a separate license agreement for Program is entered into between Accusoft and LICENSEE at that time, then the terms of that agreement and the Term of that agreement shall govern only where different from the terms and Term of this Agreement. If a separate Accusoft license agreement for Program is not entered into at that time, then LICENSEE's permitted use of Program is governed by this Paragraph 4., replacing Paragraph 3. Evaluation Mode Limited License, and all other terms and Term are according to this Agreement. In that case, Accusoft grants to LICENSEE a limited, non-exclusive, non-assignable license to install and use Program on one computer for one year

beginning on the date of purchase of Program and as may subsequently be extended by Accusoft on LICENSEE's request ("Term"). LICENSEE is only permitted to transfer this license one time to one third party provided that: a) LICENSEE does not install or use Program except on behalf of the third party, and (b) the third party also agrees to all the terms of this Agreement as LICENSEE. When LICENSEE uses PrizmDoc Cloud Service, LICENSEE's Commercial License includes a monthly transaction limit or a fixed number of transactions (known as a "Transaction Bucket") which are not time limited except as noted in Section 7 below.

5. Error and Usage Reporting. LICENSEE acknowledges that Program includes an Error and Usage Reporting mechanism that may automatically exchange error and usage information with an Accusoft server or servers over the Internet when a connection to the Internet is available.

6. Ownership. LICENSEE acknowledges and agrees that Accusoft owns all right, title and interest in the Program, in all forms, including without limitation any and all worldwide proprietary rights therein, including but not limited to trademarks, copyrights, patent rights, patent continuations, trade secrets and confidential information.

7. LICENSEE Service Restrictions

a. If LICENSEE's usage of Service exceeds their monthly transaction limit, LICENSEE agrees that Accusoft may charge LICENSEE an additional monthly fee for overage transactions at the same per transaction rate reached at their transaction limit.

b. LICENSEE agrees to indemnify and hold Accusoft harmless from any claim, action or proceeding arising in any way from LICENSEE's uploaded content or from LICENSEE's usage of uploaded content.

c. LICENSEE agrees to access or use Service solely via their licensed usage of Program.

d. LICENSEE agrees the Service account key allowing their licensed access to Service is Proprietary Information of Accusoft as defined below, that they are responsible for the security of Service account key, that they will only allow use of Service account key from their licensed usage of Program installed on their one allowed licensed host and that they will immediately notify Accusoft if they learn that their Service account key was used by any other party in any other way.

e. LICENSEE is prohibited from reverse-engineering or hacking Service including Service API's (Application Programming Interfaces).

f. LICENSEE is prohibited from removing or obscuring Accusoft or PrizmDoc marks.

g. LICENSEE agrees they will access or use Service only during Term and LICENSEE agrees to discontinue all use of Service following Term and following termination of Agreement for any other reason.

h. Transactions purchased in a Transaction Bucket do not expire based on time, except Accusoft reserves the right with thirty days notice to cancel unused transactions in the case of zero activity for 180 days, or if older than one year, or immediately if Service account is closed or terminated per the terms of this Agreement.

8. Accusoft Service Obligations

a. Accusoft will utilize its best efforts to ensure that Service is available to LICENSEE at all times.

b. All data communication between Program front-end and the Program Service back-end can be encrypted via HTTPS protocol.

c. Uploaded content and cached converted content is encrypted while it resides on the Service host filesystem.

d. Accusoft will limit access to Service hosts to necessary Accusoft employees.

9. Other Restrictions and Reservations. All rights and licenses not expressly granted to LICENSEE are reserved to Accusoft. LICENSEE shall not disassemble, decompile, decrypt or reverse engineer (except reverse engineering for the purpose of debugging modifications made by LICENSEE to LGPL-licensed portions of the Program) the Program or in any manner attempt to discover or reproduce the source code or any other copyrightable aspect of the Program, or any portion thereof. With an Evaluation Mode Limited License:

a. LICENSEE is strictly prohibited from reproducing, copying, marketing, selling, distributing, licensing, sublicensing, leasing, timesharing or renting the Program or any component thereof, and

b. LICENSEE is strictly prohibited from any commercial use of the Program, and such actions are expressly prohibited, and

c. LICENSEE is strictly prohibited from incorporating or including the Program or any component thereof, in whole or part, into or as part of any product or service of LICENSEE regardless of functionality of Program (or lack thereof) within or as part of such product or service, and

d. LICENSEE is strictly prohibited from using the Program, directly or indirectly, in developing LICENSEE's own

product with, or including, similar functionality, and

e. LICENSEE is strictly prohibited from making any copies of the Program for any purpose whatsoever.

10. Warranty Disclaimer. LICENSEE ACKNOWLEDGES AND AGREES THAT THE PROGRAM IS PROVIDED "AS IS." ACCUSOFT DISCLAIMS ANY AND ALL REPRESENTATIONS AND WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND AGAINST INFRINGEMENT.

11. Limitation of Liability. ACCUSOFT SHALL HAVE NO LIABILITY TO LICENSEE, LICENSEE AFFILIATES, SUBSIDIARIES, SHAREHOLDERS, OFFICERS, DIRECTORS, EMPLOYEES, REPRESENTATIVES OR ANY THIRD PARTY, WHETHER IN CONTRACT, TORT, NEGLIGENCE OR PRODUCTS LIABILITY, FOR ANY CLAIM, LOSS OR DAMAGE, INCLUDING BUT NOT LIMITED TO, LOST PROFITS, LOSS OF USE, BUSINESS INTERRUPTION, LOST DATA, LOST FILES, OR FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY KIND OR NATURE WHATSOEVER ARISING OUT OF OR IN CONNECTION WITH USE OF OR INABILITY TO USE THE PROGRAM, OR THE PERFORMANCE OR OPERATION OF THE PROGRAM, EVEN IF ACCUSOFT HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

12. Indemnification by LICENSEE. LICENSEE SHALL INDEMNIFY, HOLD HARMLESS AND DEFEND ACCUSOFT FOR ANY LOSS, CLAIM, ACTION OR PROCEEDING THAT ARISES OR RESULTS FROM ANY ACTIONS OR OMISSIONS OF LICENSEE PERTAINING TO THE PRODUCT OR FROM LICENSEE USAGE OF PROGRAM NOT PERMITTED BY THIS AGREEMENT.

13. Termination. This Agreement and the limited license shall expire at midnight on the last day of the Term. This Agreement shall also terminate immediately upon LICENSEE'S breach of any provision of this Agreement. Upon expiration or termination of the Term or any other termination, LICENSEE shall have no license or rights whatsoever in or regarding the Program, shall immediately cease to use the Program, and shall uninstall the Program from LICENSEE'S and any other computers, and shall destroy all copies of the Program, unless LICENSEE has entered into a separate Accusoft license agreement for the Program signed by an authorized representative of Accusoft. In the event of any termination for any reason all sections of this Agreement survive except Paragraphs 2, 3, and 4.

14. Confidentiality. LICENSEE acknowledges that the Program contains Accusoft know-how, confidential and trade secret information ("Proprietary Information"). LICENSEE agrees: (a) to hold the Proprietary Information in the strictest confidence, (b) not to, directly or indirectly, copy, reproduce, distribute, manufacture, duplicate, reveal, report, publish, disclose, cause to be disclosed, or otherwise transfer the Proprietary Information to any third party, (c) not to make use of the Proprietary Information other than for usage of Program as permitted by this Agreement and (d) to disclose the Proprietary Information only to LICENSEE'S representatives requiring such material for effective performance of this Agreement and who have undertaken an obligation of confidentiality and limitation of use consistent with this Agreement. This obligation shall continue as long as allowed under applicable law.

15. Injunctive Relief. LICENSEE agrees that any violation or threat of violation of this Agreement will result in irreparable harm to Accusoft for which damages would be an inadequate remedy. Therefore, in addition to its rights and remedies available at law (including but not limited to the recovery of damages for breach of this Agreement), Accusoft shall be entitled to immediate injunctive relief to prevent any violation of Accusoft's copyright, trademark, trade secret rights regarding the Program, or any violation of this Agreement, including, but not limited to, unauthorized use, copying, distribution or disclosure of or regarding the Program, as well as any other equitable relief as the court may deem proper under the circumstances.

16. Liquidated Damages. In the event LICENSEE other than as granted by this Agreement and other than granted by a separate Accusoft license agreement for Program (a) copies the Program, (b) uses the Program for any reason other than the Purpose, (c) installs or uses the program on more than a single computer or (d) otherwise violates or breaches this Agreement or separate Accusoft license agreement for Program, LICENSEE agrees that Accusoft is entitled to obtain as liquidated damages and not as a penalty the then current published quantity one list price for each unlicensed copy of Program distributed, copied or installed other than as granted by this Agreement or other Accusoft license agreement for Program. THE LICENSEE EXPRESSLY AGREES THAT THE FOREGOING LIQUIDATED DAMAGES ARE NOT A PENALTY.

17. No Reduced Pricing. In any determination of Accusoft's damages (whether liquidated damages or actual damages), or any determination of any licensing fees or royalties due Accusoft under this Agreement due to a

breach by LICENSEE hereunder, LICENSEE shall not be entitled to any discounts (volume or otherwise) or reduced licensing fees or royalties. Further, LICENSEE agrees that it shall not be entitled to reduced licensing fees or royalties when determining Accusoft's damages due to any undertaking or activity by LICENSEE regarding the Program outside of or exceeding the scope of permission or Purpose of this Agreement, or LICENSEE's actions otherwise in violation of this Agreement, other than as may be granted by a separate Accusoft license agreement for Program.

18. Attorneys' Fees and Costs. In the event of any lawsuit or other proceeding brought as a result of any actual or alleged breach of this Agreement, to enforce any provisions of this Agreement, or to enforce any intellectual property or other rights in or pertaining to the Program, the prevailing party shall be entitled to an award of its reasonable attorneys' fees and costs, including the costs of any expert witnesses, incurred at all levels of proceedings.

19. Governing Law. This Agreement shall be construed, governed and enforced in accordance with the laws of the State of Florida, without regard to any conflicts of laws rules. Any action related to or arising out of this Agreement will be brought solely in the state court sitting in Hillsborough County, Florida or in the federal courts in the Middle District of Florida, Tampa Division, and LICENSEE consents to the exclusive jurisdiction and venue of said courts.

20. Severability. If any provision of this Agreement is determined to be invalid by any court of final jurisdiction, then it shall be omitted and the remainder of the Agreement shall continue to be binding and enforceable. In addition, the Court is hereby authorized to enforce any provision of the Agreement that the Court otherwise deems unenforceable, to whatever lesser extent the Court deems reasonable and appropriate, rather than invalidating the entire provision. Without limiting the generality of the foregoing, LICENSEE expressly agrees that should LICENSEE be found to have breached the Agreement, under no circumstances shall LICENSEE be entitled to any volume or other discount, or reduced licensing fee or royalty in the determination of Accusoft's damages, or otherwise in the determination of any licensing fee or royalty owed to Accusoft.

21. Government Rights. The Program and accompanying documentation have been developed at private expense and are sold commercially. They are provided under any U.S. government contracts or subcontracts with the most restricted and the most limited rights permitted by law and regulation. Whenever so permitted, the government and any intermediaries will obtain only those rights specified in Accusoft's standard commercial license. Thus, the Program referenced herein, and the documentation provided by Accusoft hereunder, which are provided to any agency of the U.S. Government or U.S. Government contractor or subcontractor at any tier shall be subject to the maximum restrictions on use as permitted by FAR 52.227-19 (June 1987) or DFARS 227.7202-3(a) (Jan. 1, 2000) or successor regulations. Manufacturer is Accusoft Corporation, 4001 N. Riverside Drive Tampa, FL 33603.

22. Entire Agreement. This Agreement represents the entire understanding of the parties concerning the subject matter hereof and supersedes all prior communications and agreements, whether oral or written, relating to the subject matter of this Agreement. Only a writing signed by the parties may modify this Agreement. In the event of any modification in writing, of this Agreement, including an expanded Accusoft license agreement for Program, all unmodified, non-conflicting sections of this Agreement survive.

23. Contact Us. Should you have any questions concerning this Agreement, or if you need to modify this Agreement, or if you have an Evaluation Mode Limited License and you need to use Program for a different purpose than Purpose such as a commercial purpose, or if you desire to contact Accusoft for any other question or reason, please contact Accusoft at 1-813-875-7575 or at sales@accusoft.com.

24. Third Party Notices. See <https://help.accusoft.com/PrizmDoc/v13.17/HTML/third-party-attributions.html>

Agreement Version: 2021-03-31

Third-Party Attributions

The following third-party software may be used or distributed in the backend components (Windows PrizmDoc

Server, Windows PAS, Linux PrizmDoc Server, and/or Linux PAS), the Viewer, or the legacy samples of the Program:

- Backend: PrizmDoc Server
 - Windows Fonts
 - Linux Fonts
 - Prizmdoc Server Docker Image
- PrizmDoc Application Services (PAS)
- Viewer
 - Viewer Fonts
 - @prizmdoc/viewer-core
- PrizmDoc Viewer Eval Docker Image
- Legacy Samples

Backend: PrizmDoc Server

Active-Directory-Object-Picker

Copyright (c) 2004 by Armand du Plessis and is now extended and maintained by Tulpep

Download: <https://github.com/Tulpep/Active-Directory-Object-Picker>

License: <https://github.com/Tulpep/Active-Directory-Object-Picker/blob/master/LICENSE>

ajv

The MIT License (MIT)

Copyright (c) 2015-2017 Evgeny Poberezkin

Download: <https://www.npmjs.com/package/ajv>

License: <https://github.com/epoberezkin/ajv/blob/master/LICENSE>

Apache PDFBox (<http://pdfbox.apache.org/>)

Copyright (c) 2002-2010 The Apache Software Foundation

Download: <http://pdfbox.apache.org/download.html>

License: <http://www.apache.org/licenses/LICENSE-2.0>

Apache FontBox (<http://pdfbox.apache.org/>)

Copyright (c) 2008-2010 The Apache Software Foundation

Download: <http://pdfbox.apache.org/download.html>

License: <http://www.apache.org/licenses/LICENSE-2.0>

Apache JempBox (<http://pdfbox.apache.org/>)

Copyright (c) 2008-2010 The Apache Software Foundation

Download: <http://pdfbox.apache.org/download.html>

License: <http://www.apache.org/licenses/LICENSE-2.0>

Apache POI (<http://poi.apache.org/>)

Copyright (c) 2001-2007 The Apache Software Foundation

Download: <http://www.apache.org/dyn/closer.cgi/poi/>

License: <http://www.apache.org/licenses/LICENSE-2.0>

Apache Commons FileUpload (<http://commons.apache.org/fileupload/>)

Copyright (c) 2002-2008 The Apache Software Foundation
Download: http://commons.apache.org/fileupload/download_fileupload.cgi
License: <http://www.apache.org/licenses/LICENSE-2.0>

Apache Commons IO (<http://commons.apache.org/io/>)

Copyright (c) 2001-2008 The Apache Software Foundation
Download: http://commons.apache.org/io/download_io.cgi
License: <http://www.apache.org/licenses/LICENSE-2.0>

Apache Commons Lang (<http://commons.apache.org/lang/>)

Copyright (c) 2001-2010 The Apache Software Foundation
Download: http://commons.apache.org/lang/download_lang.cgi
License: <http://www.apache.org/licenses/LICENSE-2.0>

Apache Commons Logging (<http://commons.apache.org/logging/>)

Copyright (c) 2003-2007 The Apache Software Foundation
Download: http://commons.apache.org/logging/download_logging.cgi
License: <http://www.apache.org/licenses/LICENSE-2.0>

Apache Commons Text (<http://commons.apache.org/proper/commons-text/>)

Copyright (c) 2001-2017 The Apache Software Foundation
Download: http://commons.apache.org/proper/commons-text/download_text.cgi
License: <http://www.apache.org/licenses/LICENSE-2.0>

Apache JAMES Mime4j (<http://james.apache.org/mime4j/>)

Copyright (c) 2004-2010 The Apache Software Foundation
Download: <http://james.apache.org/download.cgi>
License: <http://www.apache.org/licenses/LICENSE-2.0>

Apache Tika (<https://tika.apache.org/>)

Copyright 2015 The Apache Software Foundation
Download: <https://tika.apache.org/download.html>
License: <https://github.com/apache/tika/blob/1.23/LICENSE.txt>

Apache Xerces (<http://xerces.apache.org/xerces-c/>)

Copyright (c) 1999-2010 The Apache Software Foundation
Download: <http://xerces.apache.org/xerces-c/download.cgi>
License: <http://www.apache.org/licenses/LICENSE-2.0>

aws-sdk-js

Apache License Version 2.0
Copyright (c) 2012-2015 Amazon.com, Inc. or its affiliates. All Rights Reserved.
Download: <https://github.com/aws/aws-sdk-js>
License: <http://www.apache.org/licenses/LICENSE-2.0>

biweekly

Copyright (c) 2013-2018, Michael Angststadt

Download: <https://github.com/mangstadt/biweekly> License:
<https://github.com/mangstadt/biweekly/blob/0.6.3/LICENSE>

body-parser (<https://github.com/expressjs/body-parser>)

Copyright (c) 2014 Jonathan Ong <me@jongleberry.com>
Copyright (c) 2014-2015 Douglas Christopher Wilson <doug@somethingdoug.com>
Download: <https://github.com/expressjs/body-parser>
License: <https://github.com/expressjs/body-parser/blob/master/LICENSE>

Boost (<http://www.boost.org>)

Download: <http://sourceforge.net/projects/boost/files/boost/1.55.0/>
Version: 1.55.0
License: <install directory>\licenses\boost\LICENSE_1_0.txt

cairo (<https://cairographics.org>)

Copyright (c) 2002 University of Southern California
Copyright (c) 2005 Red Hat, Inc.
Download: <https://www.cairographics.org/releases/>
License (LGPL v2.1): <install directory>\licenses\cairo\COPYING-LGPL-2.1

caolan/async

The MIT License (MIT)
Copyright (c) 2010-2018 Caolan McMahon
Download: <https://github.com/caolan/async>
License: <install directory>\licenses\async\LICENSE

CmdParser

The MIT License (MIT)
Copyright (c) 2015 - 2016 Florian Rapp
Download: <https://github.com/FlorianRappl/CmdParser>
License: <https://github.com/FlorianRappl/CmdParser/blob/v1.0.0/LICENSE>

compare-version

The MIT License (MIT)
(c) Kevin Mårtensson
Download: <https://www.npmjs.com/package/compare-version>
License: <https://www.npmjs.com/package/compare-version#license>

Consul

Mozilla Public License, version 2.0
Copyright (c) 2014-2018 HashiCorp, Inc.
Download: <https://github.com/hashicorp/consul>
License: <install directory>/license/Consul/LICENSE

css-color-names

The MIT License (MIT)
Copyright 2018 Dave Eddy dave@daveeddy.com
Download: <https://github.com/bahamas10/css-color-names>

License: <https://github.com/bahamas10/css-color-names#license>

Duration.js

Copyright (c) 2013 Ilia Choly

Download: <https://github.com/icholy/Duration.js>

License: MIT <https://github.com/icholy/Duration.js/blob/master/LICENSE>

expat - 2.1.0 (<http://www.libexpat.org/>)

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd and Clark Cooper

Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006 Expat maintainers

Download: <http://sourceforge.net/projects/expat/files/expat>

License: `<install directory>\licenses\expat\COPYING`

express (<http://expressjs.com/>)

Copyright (c) 2009-2014 TJ Holowaychuk <tj@vision-media.ca>

Copyright (c) 2013-2014 Roman Shtylman <shtylman+expressjs@gmail.com>

Copyright (c) 2014-2015 Douglas Christopher Wilson <doug@somethingdoug.com>

Download: <https://github.com/strongloop/express>

License: <https://github.com/strongloop/express/blob/master/LICENSE>

ffi

The MIT License (MIT)

Copyright (c) 2009-2011 Richard "Rick" W. Branson

Copyright (c) 2012-2014 Nathan Rajlich, Richard "Rick" W. Branson

Copyright (c) 2015 Nathan Rajlich, Richard "Rick" W. Branson, Gabor Mezo

Download: <https://www.npmjs.com/package/ffi>

License: <https://github.com/node-ffi/node-ffi/blob/master/LICENSE>

follow-redirects

The MIT License (MIT)

Copyright 2014–present Olivier Lalonde <olalonde@gmail.com>, James Talmage <james@talmage.io>, Ruben Verborgh

Download: <https://www.npmjs.com/package/follow-redirects>

License: <https://github.com/olalonde/follow-redirects/blob/master/LICENSE>

fontconfig - 2.11.1 (<http://www.freedesktop.org/wiki/Software/fontconfig/>)

Copyright (c) 2001, 2003 Keith Packard

Download: <http://www.freedesktop.org/software/fontconfig/release/>

License: `<install directory>\licenses\fontconfig\COPYING`

freetype - 2.5.5

Copyright (c) 2012 The FreeType Project (www.freetype.org). All rights reserved.

Download: <https://sourceforge.net/projects/freetype/files/>

License: `<install directory>\licenses\freetype\FTL.TXT`

fs-extra

The MIT License (MIT)

Copyright (c) 2011-2017 JP Richardson

Download: <https://www.npmjs.com/package/fs-extra>

License: <https://github.com/jprichardson/node-fs-extra/blob/master/LICENSE>

Glib (<https://developer.gnome.org/glib/>)

Authors: <install directory>/licenses/glib/AUTHORS

Download: <http://ftp.gnome.org/pub/gnome/sources/glib/>

License: <install directory>/licenses/glib/COPYING

HarfBuzz (<http://www.freedesktop.org/wiki/Software/HarfBuzz/>)

Copyrights: <install directory>/licenses/harfbuzz/COPYING

Download: <http://www.freedesktop.org/software/harfbuzz/release/>

License: <install directory>/licenses/harfbuzz/COPYING

http-signature

The MIT License (MIT)

Copyright Joyent, Inc. All rights reserved.

Download: <https://github.com/joyent/node-http-signature/>

License: <install directory>\licenses\http-signature\LICENSE

ICU4J - (*Windows Server Only*)

Copyright (c) 1995-2013 International Business Machines Corporation and others

Download: <http://site.icu-project.org/download>

License: <install directory>/licenses/ICU/license.htm

Java Advanced Imaging Codec

Copyright (c) 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Download: <https://maven.geomajas.org/javax/media/jai-codec/1.1.3/jai-codec-1.1.3.jar>

License: <install directory>/licenses/Sun JAI/LICENSE-jai.txt

Java Advanced Imaging API

Copyright (c) 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Download: https://repo1.maven.org/maven2/javax/media/jai_core/1.1.3/

License: http://download.java.net/media/jai/builds/release/1_1_3/LICENSE-jai.txt

Java Advanced Imaging Image I/O Tools

Copyright (c) 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Download: https://repo1.maven.org/maven2/com/sun/media/jai_imageio/1.1/

License: <http://www.opensource.org/licenses/bsd-license.php>

Java DogStatsD Client

The MIT License (MIT)

Copyright (c) 2012 youDevisе, Ltd.

Download: <https://github.com/DataDog/java-dogstatsd-client>

License: <https://github.com/DataDog/java-dogstatsd-client/blob/master/LICENSE>

JDOM

This product includes software developed by the JDOM Project (<http://www.jdom.org/>).

Copyright (c) 2000-2004 Jason Hunter & Brett McLaughlin. All rights reserved.

Source: <http://www.jdom.org/downloads/source.html>

Download: <http://www.jdom.org/downloads/index.html>

License: `<install directory>\licenses\jdom\LICENSE.txt` (<http://www.jdom.org/docs/faq.html#a0030>)

jimp

The MIT License (MIT)

Copyright (c) 2014 Oliver Moran

Download: <https://www.npmjs.com/package/jimp>

License: <https://github.com/oliver-moran/jimp/blob/master/LICENSE>

JPedal JBIG2 Image Decoder (<http://jpedaljbig2imag.sourceforge.net/>)

Copyright (c) 1997-2008, IDR solutions and Contributors.

Download: <http://sourceforge.net/projects/jpedaljbig2imag/files/>

License: BSD License (`<install directory>/licenses/jbig2_1.4/license.txt`)

OpenJDK 1.8.0.282 https://adoptopenjdk.net/release_notes.html

GNU General Public License, version 2, with the Classpath Exception

Download: https://github.com/AdoptOpenJDK/openjdk8-upstream-binaries/releases/download/jdk8u282-b08/OpenJDK8U-sources_8u282b08.tar.gz

License: `<install directory>/java/jre8/LICENSE` (<https://openjdk.java.net/legal/gplv2+ce.html>)

js-yaml (<https://github.com/nodeca/js-yaml>)

Copyright (c) 2011-2015 by Vitaly Puzrin

Download: <https://github.com/nodeca/js-yaml>

License: <https://github.com/nodeca/js-yaml/blob/master/LICENSE>

JSON.NET (<http://www.newtonsoft.com/json>) - (*Windows Server Only*)

Copyright (c) 2007 James Newton-King

Download: <http://www.newtonsoft.com/json>

License: <https://github.com/JamesNK/Newtonsoft.Json/blob/master/LICENSE.md>

jsoup

Copyright (c) 2009 - 2017 Jonathan Hedley (jonathan@hedley.net), MIT, v1.8.3

Download: <https://repo.maven.apache.org/maven2/org/jsoup/jsoup/>

License: <https://jsoup.org/license>

JTNEF (<http://www.freeutils.net/source/jtnef/>)

The JTNEF package used in this product is copyright (c) 2003-2010 by Amichai Rothman.

jutf7 (<http://jutf7.sourceforge.net/>)

The MIT License (MIT)

Copyright (c) 2006,2008 J.T. Beetstra

Download: <https://sourceforge.net/projects/jutf7/files/jutf7/1.0.0/>

License: <https://sourceforge.net/p/jutf7/code/HEAD/tree/tags/jutf7-1.0.0/LICENSE.txt>

lcms

The MIT License (MIT)

Copyright (c) 1998-2011 Marti Maria Saguer

Download: <https://sourceforge.net/projects/lcms/files/lcms/2.6/lcms2-2.6.tar.gz/download>

License: <install directory>/licenses/lcms/COPYING

libffi (<https://sourceware.org/libffi>)

Copyright (c) 1996-2014 Anthony Green, Red Hat Inc. and others

Download: <https://github.com/libffi/libffi>

License: <install directory>/licenses/libffi/LICENSE

libgif4

Copyright (c): <https://sourceforge.net/projects/giflib/files/>

Authors: <https://sourceforge.net/projects/giflib/files/>

Download: <https://sourceforge.net/projects/giflib/files/>

License: <https://sourceforge.net/projects/giflib/files/>

libintl (<https://www.gnu.org/software/gettext>)

Authors: <install directory>/licenses/libintl/AUTHORS, <install directory>/licenses/libintl/THANKS

Download: <http://ftp.gnu.org/pub/gnu/gettext/>

License: <install directory>/licenses/libintl/COPYING

libjpeg

This software is copyright (c) 1991-1998, Thomas G. Lane.

License: <install directory>\licenses\libjpeg\license.txt

libjpeg-turbo (<http://www.libjpeg-turbo.org>)

Copyright (c) 1991-2012, Thomas G. Lane, Guido Vollbeding

Download: <https://sourceforge.net/projects/libjpeg-turbo/files/>

License: <install directory>/licenses/libjpeg-turbo/README

License: <install directory>/licenses/libjpeg-turbo/README-turbo.txt

libopenjpeg2 (<http://www.openjpeg.org/>)

Authors: <http://mirrors.kernel.org/ubuntu/pool/universe/o/openjpeg/>

Copyright (c): <http://mirrors.kernel.org/ubuntu/pool/universe/o/openjpeg/>

Download: <http://mirrors.kernel.org/ubuntu/pool/universe/o/openjpeg/>

License: <http://mirrors.kernel.org/ubuntu/pool/universe/o/openjpeg/>

Libpng - 1.6.16

Copyright (c) 1998-2011 Glenn Randers-Pehrson

Download: <https://sourceforge.net/projects/libpng/files/>

License: <install directory>\licenses\libpng\LICENSE

LibreOffice (<https://www.libreoffice.org/>)

Publisher: The Document Foundation & PortableApps.com (John T. Haller)

License: LibreOffice is licensed under the GNU Lesser General Public License (LGPLv3).

Download: <https://www.libreoffice.org/download/>

Please contact [Accusoft support](#) about the LibreOffice source code distribution with modifications by Accusoft.

libtiff - 4.0.3

Copyright (c) 1988-1997 Sam Leffler

Copyright (c) 1991-1997 Silicon Graphics, Inc.

Download: <http://download.osgeo.org/libtiff/>

License: `<install directory>\licenses\libtiff\COPYRIGHT`

load-bmfont

The MIT License (MIT) Copyright (c) 2015 Jam3

Download: <https://github.com/Jam3/load-bmfont>

License: <https://github.com/Jam3/load-bmfont/blob/master/LICENSE.md>

lodash

The MIT License (MIT)

Copyright (c) JS Foundation and other contributors [<https://js.foundation/>](https://js.foundation/)

Copyright (c) OpenJS Foundation and other contributors [<https://openjsf.org/>](https://openjsf.org/)

Download: <https://github.com/lodash/lodash>

License: `<install directory>\licenses\lodash\LICENSE`

lodash-node

Copyright jQuery Foundation and other contributors [<https://jquery.org/>](https://jquery.org/)

Download: <https://www.npmjs.com/package/lodash-node>

License: <https://github.com/lodash-archive/lodash-node/blob/master/LICENSE>

log4j - Apache 2.0, v2.8

Copyright (c) 2017 The Apache Software Foundation, Licensed under the Apache License, Version 2.0.

Download: <https://repo.maven.apache.org/maven2/org/apache/logging/log4j/log4j-api/2.8/>

License: <https://www.apache.org/licenses/LICENSE-2.0>

merge-stream

The MIT License (MIT)

Copyright (c) Stephen Sugden me@stephensugden.com (stephensugden.com)

Download: <https://github.com/grncdr/merge-stream>

License: <https://github.com/grncdr/merge-stream/blob/master/LICENSE>

mime - (*Linux Server Only*)

The MIT License (MIT)

Copyright (c) 2010 Benjamin Thomas, Robert Kieffer

Download: <https://github.com/broofa/node-mime>

License: License: `<install directory>\licenses\mime\LICENSE`

mkdirp

The MIT License (MIT)

Copyright (c) 2010 James Halliday (mail@substack.net)

Download: <https://github.com/substack/node-mkdirp>

License: License: `<install directory>\licenses\mkdirp\LICENSE`

MongoDB

Free Software Foundation's GNU AGPL v3.0

Copyright (c) 2016 MongoDB, Inc.

Download: <https://www.mongodb.com/download-center#community>

License: <https://www.gnu.org/licenses/agpl-3.0.html>

Server Side Public License VERSION 1, OCTOBER 16, 2018

Copyright (c) 2018 MongoDB, Inc.

Download: <https://www.mongodb.com/try/download/community>

License: <https://www.mongodb.com/licensing/server-side-public-license>

Mono - (*Linux Server Only*)

Download: <http://www.mono-project.com/download/stable/>

License(s): `<install directory>\licenses\Mono\LICENSE`

moment

The MIT License (MIT)

Copyright (c) JS Foundation and other contributors

Download: <https://github.com/moment/moment/tree/2.29.1>

License: <https://github.com/moment/moment/blob/2.29.1/LICENSE>

moment-timezone

The MIT License (MIT)

Copyright (c) JS Foundation and other contributors

Download: <https://github.com/moment/moment-timezone/>

License: <https://github.com/moment/moment-timezone/blob/develop/LICENSE>

mv - 2.1.1

The MIT License (MIT)

Copyright (c) 2014 Andrew Kelley

Download: <https://github.com/andrewrk/node-mv>

License: `<install directory>\licenses\mv\LICENSE`

nan

The MIT License (MIT)

Copyright (c) 2018 NAN contributors

Download: <https://www.npmjs.com/package/nan>

License: <https://github.com/nodejs/nan/blob/master/LICENSE.md>

netty-buffer,

netty-codec,

netty-codec-http,

netty-common,

netty-handler,

netty-transport:

Apache License, Version 2.0

Copyright (c) 2012 The Netty Project

Download: <https://github.com/netty/netty/>

License: <https://github.com/netty/netty/blob/4.1/LICENSE.txt>

Ninject

Copyright (c) 2007-2012 Enkari, Ltd and the Ninject project contributors

Download: <http://www.ninject.org/download.html>

License: `<install directory>/licenses/Ninject/LICENSE.txt`

(<http://www.apache.org/licenses/LICENSE-2.0>)

NLog

Copyright (c) 2004-2016 Jaroslaw Kowalski jaak@jkowalski.net, Kim Christensen, Julian Verdurmen

Download: <http://nlog-project.org/download/>

License: `<install directory>/licenses/NLog/LICENSE.txt`

node-assert-plus (<https://github.com/mcavage/node-assert-plus>)

The MIT License (MIT)

Copyright (c) 2012 Mark Cavage

Download: <https://github.com/mcavage/node-assert-plus>

License: `<install directory>\licenses\assert-plus\LICENSE`

node-bunyan

The MIT License (MIT)

Copyright (c) 2011-2012 Joyent Inc.

Download: <https://github.com/trentm/node-bunyan>

License: `<install directory>\licenses\bunyan\LICENSE`

node-consul

The MIT License (MIT)

Copyright (c) 2014 Silas Sewell

Download: <https://github.com/silas/node-consul>

License: `<install directory>/licenses/node-consul/LICENSE`

node-gyp

The MIT License (MIT)

Copyright (c) 2012 Nathan Rajlich nathan@tootallnate.net

Download: <https://www.npmjs.com/package/node-gyp>

License: <https://github.com/nodejs/node-gyp/blob/master/LICENSE>

node-mongodb-native

Apache License, Version 2.0, January 2004

Download: <https://github.com/mongodb/node-mongodb-native>

License: <https://github.com/mongodb/node-mongodb-native/blob/2.2/LICENSE>

node-mysql2

The MIT License (MIT)

Copyright (c) 2016 Andrey Sidorov (sidorares@yandex.ru) and contributors

Download: <https://github.com/sidorares/node-mysql2>

License: <install directory>/licenses/node-mysql2/License

node-remove

The MIT License (MIT)

Download: <https://github.com/dsc/node-remove>

License: <install directory>/licenses/node-remove/package.json

node-retry

Copyright: (c) 2011:

Tim Koschützki (tim@debuggable.com)

Felix Geisendörfer (felix@debuggable.com)

download: <https://github.com/tim-kos/node-retry>

license: MIT <https://github.com/tim-kos/node-retry/blob/master/License>

node-schedule

The MIT License (MIT)

Copyright (C) 2015 Matt Patenaude

Download: <https://www.npmjs.com/package/node-schedule>

License: <https://github.com/node-schedule/node-schedule/blob/master/LICENSE>

node-statsd

The MIT License (MIT)

Copyright 2011 Steve Ivy. All rights reserved.

Download: <https://www.npmjs.com/package/node-statsd>

License: <https://github.com/sivy/node-statsd/blob/master/LICENSE>

node-stream

The MIT License (MIT)

Copyright (c) 2016 Stephen Zuniga

Download: <https://github.com/stezu/node-stream>

License: <https://github.com/stezu/node-stream/blob/master/LICENSE>

node-stream-meter

The MIT License (MIT)

Copyright (c) Bryce B. Baril <bryce@ravenwall.com>

Download: <https://github.com/brycebaril/node-stream-meter>

License: <install directory>/licenses/node-stream-meter/LICENSE

node-uuid

The MIT License (MIT)

Copyright (c) 2010-2012 Robert Kieffer

Download: <https://github.com/broofa/node-uuid>

License: <install directory>/licenses/node-uuid/LICENSE

Node.js v4.4.5

Copyright (c) Node.js contributors. All rights reserved.

Download: <http://nodejs.org/dist/v4.4.5/>

License: <install directory>\licenses\node.js 4\LICENSE

Node.js v8.9.0

Copyright (c) Node.js contributors. All rights reserved.

Download: <http://nodejs.org/dist/v8.9.0/>

License: <install directory>\licenses\node.js 8\LICENSE

Node.js v10.15.3

Copyright (c) Node.js contributors. All rights reserved.

Download: <http://nodejs.org/dist/v10.15.3/>

License: <install directory>\licenses\node.js 10\LICENSE

npm

The Artistic License 2.0

Copyright (c) npm, Inc. and Contributors

Download: <https://www.npmjs.com/package/npm>

License: <https://github.com/npm/cli/blob/latest/LICENSE>

once

ISC license

Copyright (c) Isaac Z. Schlueter and Contributors

Download: <https://github.com/isaacs/once>

License: <install directory>\licenses\once\LICENSE

OpenJPEG library (<http://www.openjpeg.org>)

Copyrights: <install directory>/licenses/openjpeg/LICENSE

Download: <https://github.com/uclouvain/openjpeg/releases/>

License: <install directory>/licenses/openjpeg/LICENSE

org-everit/json-schema (<https://github.com/everit-org/json-schema>)

Apache License Version 2.0

Version 2.0, January 2004

Download: <https://github.com/everit-org/json-schema>

License: <http://www.apache.org/licenses/LICENSE-2.0>

Pango (<http://www.pango.org>)

Authors: <install directory>/licenses/pango/AUTHORS, <install directory>/licenses/pango/THANKS

Download: <http://ftp.gnome.org/pub/gnome/sources/pango/>

License: <install directory>/licenses/pango/COPYING

PDFOne - v5.4.877.546

Copyright (c) 2002-2018 Gnostice Information Technologies Private Limited. All rights reserved.

License: https://www.gnostice.com/PDFOne_Java.asp?show=licensing

Pixman (<http://www.pixman.org>)

Copyrights: <install directory>/licenses/pixman/COPYING

Download: <https://cairographics.org/releases/>
License: <install directory>/licenses/pixman/COPYING

pm2 (<http://pm2.keymetrics.io/>)

Copyright (c) 2013-2015 Strzelewicz Alexandre
Download: <https://github.com/Unitech/PM2>
License: <https://github.com/Unitech/pm2/blob/master/GNU-AGPL-3.0.txt>

pmx

The MIT License (MIT)
Download: <https://www.npmjs.com/package/pmx>
License: <https://github.com/keymetrics/pmx#license>

Poppler

Copyright (c) 2005-2018 The Poppler Developers
Copyright (c) 1996-2011 Glyph & Cog, LLC
Download: <http://poppler.freedesktop.org>
License: <install directory>\licenses\poppler\COPYING

Poppler-data

Copyright 1990-2009 Adobe Systems Incorporated.
Copyright Glyph & Cog, LLC
Download: <http://poppler.freedesktop.org>
Licenses: <install directory>\licenses\poppler-data

ps-node

The MIT License (MIT)
Copyright (c) 2015 Neekey
Download: <https://github.com/neekey/ps>
License: <https://github.com/neekey/ps/blob/master/LICENSE.txt>

pump

The MIT License (MIT)
Copyright (c) 2014 Mathias Buus
Download: <https://github.com/mafintosh/pump>
License: <https://github.com/mafintosh/pump/blob/master/LICENSE>

qs

BSD license
Copyright (c) 2014 Nathan LaFreniere and other contributors
Download: <https://github.com/hapijs/qs>
License: <install directory>\licenses\qs\LICENSE

rapidjson

Copyright (c) 2011 Milo Yip (miloyip@gmail.com)
Download: <https://github.com/miloyip/rapidjson>
Version: v0.11
License: <install directory>/licenses/rapidjson/license.txt

ref

The MIT License (MIT)
Copyright (c) 2012 Nathan Rajlich <nathan@tootallnate.net>
Download: <https://www.npmjs.com/package/ref>
License: <https://github.com/TooTallNate/ref#license>

ref-struct

The MIT License (MIT)
Copyright (c) 2012 Nathan Rajlich <nathan@tootallnate.net>
Download: <https://www.npmjs.com/package/ref-struct>
License: <https://github.com/TooTallNate/ref-struct#license>

reflections - v0.9.10

Download: <https://repo.maven.apache.org/maven2/org/reflections/reflections/0.9.10/>
License: <https://opensource.org/licenses/BSD-2-Clause>

request

Apache License Version 2.0
Version 2.0, January 2004
Download: <https://github.com/request/request>
License: <http://www.apache.org/licenses/>

restify

The MIT License (MIT)
Copyright (c) 2011 Mark Cavage, All rights reserved.
Download: <https://github.com/restify/node-restify>
License: <install directory>\licenses\restify\LICENSE

rimraf

ISC license
Copyright (c) Isaac Z. Schlueter and Contributors
Download: <https://github.com/isaacs/rimraf>
License: <install directory>\licenses\rimraf\LICENSE

RTF Parser Kit (<https://github.com/joniles/rtfparserkit>)

Copyright (c) 2013 Jon Iles
Download: <https://github.com/joniles/rtfparserkit>
License: Apache License Version 2.0 (<install directory>/licenses/RTF Parser Kit/licence.txt)

sequelize

The MIT License (MIT)
Copyright (c) 2014-present Sequelize contributors
Download: <https://github.com/sequelize/sequelize>
License: <install directory>/licenses/sequelize/LICENSE

statsd-csharp-client

The MIT License (MIT)

Copyright (c) 2015 Kyle West (kwest2123@yahoo.com) and all contributors

Download: <https://github.com/kyle2123/statsd-csharp-client>

License: <https://github.com/Kyle2123/statsd-csharp-client/blob/master/MIT-LICENCE.md>

String Search (<https://johannburkard.de/software/stringsearch/>)

StringSearch - high-performance pattern matching algorithms in Java

Copyright (c) 2003-2010 Johann Burkard

Download: <https://johannburkard.de/software/stringsearch/>

License: <https://johannburkard.de/software/stringsearch/copying.txt>

The Legion of the Bouncy Castle

Copyright (c) 2000-2009 The Legion Of The Bouncy Castle (<http://www.bouncycastle.org>)

Download: http://bouncycastle.org/latest_releases.html

License: <http://www.bouncycastle.org/licence.html>

Note: The Linux Server component of the Program also uses the following software, installed separately:

OpenOffice.org (<http://www.openoffice.org/>)

through2 (<https://github.com/rvagg/through2>)

Copyright (c) 2013, Rod Vagg (the "Original Author")

Download: <https://github.com/rvagg/through2>

License: <https://github.com/rvagg/through2>

Touch.exe (<https://www.codeproject.com/Articles/3258/Touch-for-Windows>)

Copyright (c) 2002 by Jorgen Sigvardsson

Download: https://www.codeproject.com/KB/applications/touch_win/touch_win_demo.zip

License: <https://www.codeproject.com/Articles/3258/Touch-for-Windows>

TRE, a regex matching library with support for approximate matching (<http://laurikari.net/tre/>)

Copyright (c) 2001-2009 Ville Laurikari <vl@iki.fi>. All rights reserved.

Download: <https://github.com/laurikari/tre/>

License: 2-clause BSD-like license (<install directory>/licenses/tre/LICENSE)

tree-kill

The MIT License (MIT)

Download: <https://github.com/pkrumins/node-tree-kill>

License: <https://spdx.org/licenses/MIT.html>

tunnel-agent

Apache License Version 2.0

Download: <https://github.com/mikeal/tunnel-agent>

License: <install directory>\licenses\tunnel-agent\LICENSE

uuid

The MIT License (MIT)

Copyright (c) 2010-2020 Robert Kieffer and other contributors Download: <https://github.com/uuidjs/uuid>

License: <https://github.com/uuidjs/uuid/blob/master/LICENSE.md>

WixWPF

Copyright (c) 2013 by Troy Palacino

Download: <http://wixwpf.codeplex.com/releases/view/615076>

License: <http://wixwpf.codeplex.com/license>

wkhtmltopdf (<http://wkhtmltopdf.org/>)

Copyright (c) 2007 Free Software Foundation, Inc. <<http://fsf.org/>>

Download: <http://wkhtmltopdf.org/downloads.html>

License: <install directory>/licenses/wkhtmltopdf/LICENSE

Uses Qt (www.qt.io) with modifications by Accusoft.

Copyright (c) 2015 The Qt Company Ltd.

License: <install directory>/licenses/wkhtmltopdf/qt/LICENSE.LGPLv3

Please contact Accusoft support at support@accusoft.com about the Qt source code distribution with modifications by Accusoft.

wrappy

ISC License

Copyright (c) Isaac Z. Schlueter and Contributors

Download: <https://github.com/npm/wrappy>

License: <install directory>\licenses\wrappy\LICENSE

ws

The MIT License (MIT)

Copyright (c) 2011 Einar Otto Stangvik <einaros@gmail.com>

Download: <https://github.com/websockets/ws>

License: <https://github.com/websockets/ws#license>

xml2js

The MIT License (MIT)

Copyright (c) 2010, 2011, 2012, 2013. All rights reserved.

Download: <https://github.com/Leonidas-from-XIV/node-xml2js>

License: <install directory>\licenses\xml2js\LICENSE

xmlbuilder-js

The MIT License (MIT)

Copyright (c) 2013 Ozgur Ozcitak

Download: <https://github.com/oozcitak/xmlbuilder-js>

License: <install directory>\licenses\xmlbuilder-js\LICENSE

xsp - (*Linux Server Only)

Copyright (c) 2002, 2003, 2004 Novell, Inc. and the individuals listed on the ChangeLog entries

License: <install directory>\licenses\xsp\COPYING

yargs

Copyright (c) 2010 James Halliday (mail@substack.net)

Download: <https://github.com/bcoe/yargs>

License: MIT/X11 <https://github.com/bcoe/yargs/blob/master/LICENSE>

zlib - 1.2.8

Copyright (c) 1995-2010 Jean-loup Gailly and Mark Adler
Download: <http://sourceforge.net/projects/libpng/files/zlib/>
License: <install directory>\licenses\zlib\README

Windows Fonts

AC Kaisyo (<https://www.ac-font.com>)

Copyright (c) by Font AC
Download: https://www.ac-font.com/jp/detail_jb_007.php
License: <https://www.ac-font.com/jp/terms.php>

Noto Sans (<https://www.google.com/get/noto/>)

Copyright (c) 2012 Google Inc. All Rights Reserved.
Download: <https://www.google.com/get/noto/>
License: <http://scripts.sil.org/OFL> (SIL Open Font License v1.1)

Noto Sans JP (<https://www.google.com/get/noto/help/cjk/>)

Copyright (c) 2014, 2015 Adobe Systems Incorporated (<http://www.adobe.com/>)
Download: <https://www.google.com/get/noto/help/cjk/>
License: <http://scripts.sil.org/OFL> (SIL Open Font License v1.1)

Noto Sans JP Accusoft (part of Noto Sans CJK modified by Accusoft)

Copyright (c) 2016 Accusoft Corporation, 2014, 2015 Adobe Systems Incorporated (<http://www.adobe.com/>)
License: <http://scripts.sil.org/OFL> (SIL Open Font License v1.1)

Linux Fonts

AC Kaisyo (<https://www.ac-font.com>)

Copyright (c) by Font AC
Download: https://www.ac-font.com/jp/detail_jb_007.php
License: <https://www.ac-font.com/jp/terms.php>

Economica (<https://www.fontsquirrel.com/fonts/economica>)

Copyright (c) 2012, Vicente Lamonaca (produccion.taller@gmail.com)
Download: <https://www.fontsquirrel.com/fonts/economica>
License: <http://scripts.sil.org/OFL> (SIL Open Font License v1.1)

FreeMono (<http://savannah.gnu.org/projects/freefont/>)

Copyright (c) 2002, 2003, 2005, 2008, 2009, 2010, 2012 GNU Freefont contributors.
Download: <http://ftp.gnu.org/gnu/freefont/>
License: <install directory>/licenses/fonts/freefont/COPYING (GPLv3)

IPAexMincho (<https://moji.or.jp/ipafont/>)

Copyright (c) IPA Information-technology Promotion Agency, Japan.
Download: <https://moji.or.jp/ipafont/ipafontdownload/>
License: <https://moji.or.jp/ipafont/license/> (IPA Font License Agreement v1.0)

Josefin Sans (<https://www.fontsquirrel.com/fonts/josefin-sans>)

Copyright (c) 2010 by Typemade. All rights reserved.

Download: <https://www.fontsquirrel.com/fonts/josefin-sans>

License: <http://scripts.sil.org/OFL> (SIL Open Font License v1.1)

Microsoft TrueType core fonts for the Web

Copyright (c) 2001 Microsoft Corporation. All rights reserved.

License: <http://corefonts.sourceforge.net/eula.htm> (TrueType core fonts for the Web EULA)

Noto Sans (<https://www.google.com/get/noto/>)

Copyright (c) 2012 Google Inc. All Rights Reserved.

Download: <https://www.google.com/get/noto/>

License: <http://scripts.sil.org/OFL> (SIL Open Font License v1.1)

Noto Sans JP (<https://www.google.com/get/noto/help/cjk/>)

Copyright (c) 2014, 2015 Adobe Systems Incorporated (<http://www.adobe.com/>)

Download: <https://www.google.com/get/noto/help/cjk/>

License: <http://scripts.sil.org/OFL> (SIL Open Font License v1.1)

Noto Sans JP Accusoft (part of Noto Sans CJK modified by Accusoft)

Copyright (c) 2016 Accusoft Corporation, 2014, 2015 Adobe Systems Incorporated (<http://www.adobe.com/>)

License: <http://scripts.sil.org/OFL> (SIL Open Font License v1.1)

TeX Gyre Adventor (<http://www.gust.org.pl/projects/e-foundry/tex-gyre/adventor>)

Copyright (c) 2007-2009 for TeX Gyre extensions by B. Jackowski and J.M. Nowacki (on behalf of TeX Users Groups). Vietnamese characters were added by Han The Thanh.

Download: <http://www.gust.org.pl/projects/e-foundry/tex-gyre/adventor>

License: `<install directory>/licenses/fonts/tex-gyre-adventor/GUST-FONT-LICENSE.txt` (GUST Font License)

TeX Gyre Adventor Accusoft (TXGAAccusoft, TeX Gyre Adventor font modified by Accusoft)

Copyright (c) 2016 Accusoft, 2007-2009 for TeX Gyre extensions by B. Jackowski and J.M. Nowacki (on behalf of TeX Users Groups).

Vietnamese characters were added by Han The Thanh.

License: `<install directory>/licenses/fonts/tex-gyre-adventor/GUST-FONT-LICENSE.txt` (GUST Font License)

TeX Gyre Pagella (<http://www.gust.org.pl/projects/e-foundry/tex-gyre/pagella>)

Copyright (c) 2007-2009 for TeX Gyre extensions by B. Jackowski and J.M. Nowacki (on behalf of TeX Users Groups). Vietnamese characters were added by Han The Thanh.

Download: <http://www.gust.org.pl/projects/e-foundry/tex-gyre/pagella>

License: `<install directory>/licenses/fonts/tex-gyre-pagella/GUST-FONT-LICENSE.txt` (GUST Font License)

TeX Gyre Pagella Accusoft (TXGAAccusoft, TeX Gyre Pagella font modified by Accusoft)

Copyright (c) 2016 Accusoft, 2007-2009 for TeX Gyre extensions by B. Jackowski and J.M. Nowacki (on behalf of TeX Users Groups).

Vietnamese characters were added by Han The Thanh.

License: `<install directory>/licenses/fonts/tex-gyre-pagella/GUST-FONT-LICENSE.txt` (GUST Font License)

PrizmDoc Server Docker Image

fontconfig - 2.11.94-0ubuntu1.1

Copyright (c) 2001, 2003 Keith Packard

Download: http://archive.ubuntu.com/ubuntu/pool/main/f/fontconfig/fontconfig_2.11.94.orig.tar.bz2

License: http://changelogs.ubuntu.com/changelogs/pool/main/f/fontconfig/fontconfig_2.11.94-0ubuntu1.1/copyright

libcups2 - 2.1.3-4ubuntu0.11

GNU General Public License, version 2 with Apple Operating System Development License Exception

GNU Library General Public License, version 2 with Apple Operating System Development License Exception

Zlib license

BSD 2-clause

Copyright (c) 2007-2015, Apple Inc.

Copyright (c) 2007-2013, Apple Inc.

Copyright (c) 2005, Easy Software Products

Copyright (c) 1999, Aladdin Enterprises.

Copyright (c) 2011, Red Hat, Inc

Copyright (c) 2007-2014, Apple Inc

Copyright (c) 1997-2007, Easy Software Products

Download: http://archive.ubuntu.com/ubuntu/pool/main/c/cups/cups_2.1.3.orig.tar.bz2

License: http://changelogs.ubuntu.com/changelogs/pool/main/c/cups/cups_2.1.3-4ubuntu0.11/copyright with references to the full text of [GPL-2.0](#) and [LGPL-2.0](#)

libdbus-glib-1-2 - 0.106-1

GNU General Public License, version 2 or any later version

Expat license

Copyright (c) 2002-2010 Red Hat, Inc

Copyright (c) 2002-2003 CodeFactory AB

Copyright (c) 2004 Ximian, Inc

Copyright (c) 2005-2011 Nokia Corporation

Copyright (c) 2006 Steve Frécinaux

Copyright (c) 2007 Codethink Ltd

Copyright (c) 2009-2014 Collabora Ltd

Copyright (c) 2013 Intel Corporation

Copyright (c) 2008 David Zeuthen

Copyright (c) 2009 Collabora Ltd

Copyright (c) 2009-2011 Nokia Corporation

Copyright (c) 2011 Nokia Corporation

Copyright (c) 2013 Collabora Ltd

Download: http://archive.ubuntu.com/ubuntu/pool/main/d/dbus-glib/dbus-glib_0.106.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/d/dbus-glib/dbus-glib_0.106-1/copyright with references to the full text of [GPL-2+](#)

libexif12 - 0.6.21-2ubuntu0.6

GNU General Public License, version 2 or any later version
GNU Lesser General Public License, version 2.1 or any later version
BSD 2-clause
Copyright (c) 2001-2009, Lutz Müller urc8@rz.uni-karlsruhe.de
Copyright (c) 2004-2009, Jan Patera patera@users.sourceforge.net
Copyright (c) 2004, Joerg Hoh joerg@devone.org
Copyright (c) 2005-2006, Hubert Figuiere hub@figuiere.net
Copyright (c) 2002-2005, Hans Ulrich Niedermann gp@n-dimensional.de
Copyright (c) 2007-2010, Dan Fandrich dan@coneharvesters.com
Copyright (c) 2002, Fredrik fredrik@krixor.xy.org
Copyright (c) 2002, Javier Achirica achirica@ttd.net
Copyright (c) 2002, Semyon Sosin sem@best.com
Copyright (c) 2002, Guido Ostkamp guido.ostkamp@t-online.de
Copyright (c) 2002, Takuro Ashie makeinu@users.sourceforge.net
Copyright (c) 2002, Jason Sodergren jason@taiga.com
Copyright (c) 2002, Renchi Raju renchi@pooh.tam.uiuc.edu
Copyright (c) 2003, Torgeir Hansen torgeir@trenger.ro
Copyright (c) 2003, Roberto Costa roberto.costa@ensta.org
Copyright (c) 2004, Angela Wrobel
Copyright (c) 2002, Basil Dias basil.dias@wipro.com
Copyright (c) 2003, Ralph Heidelberg RHeidelberg@Pinnaclesys.com
Copyright (c) 2002, Javier Achirica achirica@ttd.net
Copyright (c) 2003, Gernot Jander gernot@bigpond.com
Copyright (c) 2004, Antonio Scuri scuri@tecgraf.puc-rio.br
Copyright (c) 2002, Mark Pulford mark@kyne.com.au
Copyright (c) 2002, Semyon Sosin sem@best.com
Copyright (c) 2002, Marcus Meissner marcus@jet.franken.de
Copyright (c) 2003, Jens Finke jens@triq.net
Copyright (c) 2007, Meder Kydyraliev
Copyright (c) 2008, Mika Raento mikie@google.com
Copyright (c) 2003, Peter Bieringer pb@bieringer.de
Copyright (c) 2004, Antonio Scuri scuri@tecgraf.puc-rio.br
Copyright (c) 2008-2010, Erwin Poeze
Copyright (c) 2008-2010, Marcus Meissner
Copyright (c) 2008-2010, Jakub Bogusz
Copyright (c) 2008, Ivan Masár
Copyright (c) 2008-2010, Clytie Siddall
Copyright (c) 2009, Daniel Nylander
Copyright (c) 2009, Jan Patera
Copyright (c) 2009-2010, Marko Uskokovic
Copyright (c) 2009-2010, Tao Wei weitao1979@gmail.com
Copyright (c) 2009-2010, Vilson Gjerci vilsongjerci@gmail.com
Copyright (c) 2009, nglNx
Copyright (c) 2010, Launchpad Translators
Copyright (c) 2009, Shushi Kurose
Copyright (c) 2009-2010, Tadashi Jokagi
Copyright (c) 2009-2010, Sergio Zanchetta
Copyright (c) 2009, Bruce Cowan
Copyright (c) 2009-2010, Iryna Nikanchuk defragbrain@gmail.com
Copyright (c) 2009-2010, Joe Hansen
Copyright (c) 2010, Enes Ateş e.n.3.s@hotmail.com
Copyright (c) 2010, Launchpad Translators
Copyright (c) 2010, Robert Readman
Copyright (c) 2010, André Gondim andregondim@ubuntu.com

Copyright (c) 2010, Alexey Ivanov alexey.ivanov@gmail.com

Copyright (c) 2009, André Gondi

Copyright (c) 2002-2004, Christophe Barbe christophe@debian.org

Copyright (c) 2004-2007, Frederic Peters fpeters@debian.org

Copyright (c) 2009-2011, Emmanuel Bouthenot kolter@debian.org

Download: http://archive.ubuntu.com/ubuntu/pool/main/libe/libexif/libexif_0.6.21.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libe/libexif/libexif_0.6.21-2ubuntu0.6/copyright
with references to the full text of [GPL-2+](#) and [LGPL-2.1+](#)

libexpat1 - 2.1.0-7ubuntu0.16.04.5

The MIT License (MIT)

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd and Clark Cooper

Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006 Expat maintainers

Download: http://archive.ubuntu.com/ubuntu/pool/main/e/expat/expat_2.1.0.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/e/expat/expat_2.1.0-7ubuntu0.16.04.5/copyright

libfreetype6 - 2.6.1-0.1ubuntu2.5

GNU General Public License, version 2 or any later version

Zlib license

BSD 2-clause

BSD 3-clause

The Catharon Open Source LICENSE

The OpenGroup BSD-like license Copyright (c) 1996-2012 David Turner, Robert Wilhelm, and Werner Lemberg

Copyright (c) 1996-2009 Just van Rossum

Copyright (c) 2002-2012 Roberto Alameda

Copyright (c) 2003 Huw D M Davies for Codeweavers

Copyright (c) 2003-2012 Masatake YAMATO, Redhat K.K.

Copyright (c) 2004-2012 Albert Chin-A-Young

Copyright (c) 2004-2012 Suzuki Toshiya

Copyright (c) 2007 Dmitry Timoshkov for Codeweavers

Copyright (c) 2007-2011 Rahul Bhalariao rahul.bhalerao@redhat.com

Copyright (c) 2007-2012 Derek Clegg, Michael Toftdal

Copyright (c) 2009-2011 Oran Agra, Mickey Gabel

Copyright (c) 2010, 2012 Joel Klinghed

Copyright (c) 1996-2012 Christoph Lameter clameter@waterf.org, Anthony Fok foka@debian.org, Steve Langasek vorlon@debian.org, et al.

Copyright (c) 2001, 2002 Catharon Productions Inc.

Copyright (c) 1995-2002 Jean-loup Gailly and Mark Adler

Copyright (c) 2002-2006, 2009-2012 David Turner, Robert Wilhelm, Werner Lemberg

Copyright (c) 2005-2008 George Williams

Copyright (c) 2000-2012 Francesco Zappa Nardelli francesco.zappa.nardelli@ens.fr

Copyright (c) 2000 Computing Research Labs, New Mexico State University

Copyright (c) 1990, 1994, 1998 The Open Group

Download: http://archive.ubuntu.com/ubuntu/pool/main/f/freetype/freetype_2.6.1.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/f/freetype/freetype_2.6.1-0.1ubuntu2.5/copyright
with references to the full text of [GPL-2+](#)

libjpeg62 - 1:6b2-2

Copyright (C) 1991-1998, Thomas G. Lane.

The Graphics Interchange Format(c) is the Copyright property of CompuServe Incorporated.

GIF(sm) is a Service Mark property of CompuServe Incorporated.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libj/libjpeg6b/libjpeg6b_6b2.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/universe/libj/libjpeg6b/libjpeg6b_6b2-2/copyright

libpixman-1-0 - 0.33.6-1

The MIT License (MIT)

Copyright (c) 1987, 1988, 1989, 1998 The Open Group

Copyright (c) 1987, 1988, 1989 Digital Equipment Corporation

Copyright (c) 1999, 2004, 2008 Keith Packard

Copyright (c) 2000 SuSE, Inc.

Copyright (c) 2000 Keith Packard, member of The XFree86 Project, Inc.

Copyright (c) 2004, 2005, 2007, 2008, 2009, 2010 Red Hat, Inc.

Copyright (c) 2004 Nicholas Miell

Copyright (c) 2005 Lars Knoll & Zack Rusin, Trolltech

Copyright (c) 2005 Trolltech AS

Copyright (c) 2007 Luca Barbato

Copyright (c) 2008 Aaron Plattner, NVIDIA Corporation

Copyright (c) 2008 Rodrigo Kumpera

Copyright (c) 2008 Andr   Tupinamb  ; Copyright (c) 2008 Mozilla Corporation

Copyright (c) 2008 Frederic Plourde

Copyright (c) 2009, Oracle and/or its affiliates. All rights reserved.

Copyright (c) 2009, 2010 Nokia Corporation

Download: http://archive.ubuntu.com/ubuntu/pool/main/p/pixman/pixman_0.33.6.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/p/pixman/pixman_0.33.6-1/copyright

libx11-6 - 2:1.6.3-1ubuntu2.2

The MIT License (MIT)

Copyright (c) 2003-2006,2008 Jamey Sharp, Josh Triplett

Copyright (c) 2009 Red Hat, Inc.

Copyright (c) 1990-1992,1999,2000,2004,2009,2010 Oracle and/or its affiliates. All rights reserved.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libx11/libx11_1.6.3.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libx11/libx11_1.6.3-1ubuntu2.2/copyright

libxau6 - 1:1.0.8-1

Copyright (c) 1988, 1998 The Open Group

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libxau/libxau_1.0.8.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libxau/libxau_1.0.8-1/copyright

libxext6 - 2:1.3.3-1

Copyright (c) 1986, 1987, 1988, 1989, 1994, 1998 The Open Group

Copyright (c) 1996 Digital Equipment Corporation, Maynard, Massachusetts.

Copyright (c) 1997 by Silicon Graphics Computer Systems, Inc.

Copyright (c) 1992 Network Computing Devices

Copyright (c) 1991,1993 by Digital Equipment Corporation, Maynard, Massachusetts, and Olivetti Research Limited, Cambridge, England.

Copyright (c) 1986, 1987, 1988 by Hewlett-Packard Corporation

Copyright (c) 1994, 1995 Hewlett-Packard Company

Copyright (c) Digital Equipment Corporation, 1996

Copyright (c) 1999, 2005, 2006, Oracle and/or its affiliates. All rights reserved.

Copyright (c) 1989 X Consortium, Inc. and Digital Equipment Corporation.

Copyright (c) 1992 X Consortium, Inc. and Intergraph Corporation.

Copyright (c) 1993 X Consortium, Inc. and Silicon Graphics, Inc.

Copyright (c) 1994, 1995 X Consortium, Inc. and Hewlett-Packard Company.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libxext/libxext_1.3.3.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libxext/libxext_1.3.3-1/copyright

libxinerama1 - 2:1.1.3-1

Copyright (c) 2007, Oracle and/or its affiliates. All rights reserved.

Copyright (c) 2003 The Open Group

Copyright (c) 1991, 1997 Digital Equipment Corporation, Maynard, Massachusetts.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libxinerama/libxinerama_1.1.3.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libxinerama/libxinerama_1.1.3-1/copyright

libxrender1 - 1:0.9.9-0ubuntu1

Copyright (c) 2001,2003 Keith Packard

Copyright (c) 2000 SuSE, Inc.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libxrender/libxrender_0.9.9.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libxrender/libxrender_0.9.9-0ubuntu1/copyright

libxtst6 - 2:1.2.2-1

Copyright (c) 1990, 1991 by UniSoft Group Limited

Copyright (c) 1992, 1993, 1995, 1998 The Open Group

Copyright (c) 1995 Network Computing Devices

Copyright (c) 2005 Red Hat, Inc.

Copyright (c) 1992 by UniSoft Group Ltd.

Copyright (c) 1992, 1994, 1995 X Consortium

Copyright (c) 1994 Network Computing Devices, Inc.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libxtst/libxtst_1.2.2.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libxtst/libxtst_1.2.2-1/copyright

libssl1.0.0 - 1.0.2g-1ubuntu4.19

BSD License

Copyright (c) 1998-2004 The OpenSSL Project

Copyright (c) 1995-1998 Eric A. Young, Tim J. Hudson

Copyright (c) 1998-2004 The OpenSSL Project. All rights reserved.

Copyright (C) 1995-1998 Eric Young (ey@cryptsoft.com). All rights reserved.

Download: http://archive.ubuntu.com/ubuntu/pool/main/o/openssl/openssl_1.0.2g.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/o/openssl/openssl_1.0.2g-1ubuntu4.19/copyright

libxslt1.1 - 1.1.28-2.1ubuntu0.3

Copyright (c) 2001-2002 Daniel Veillard. All Rights Reserved.

Copyright (c) 2001-2002 Thomas Broyer, Charlie Bozeman and Daniel Veillard. All Rights Reserved.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libxslt/libxslt_1.1.28.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libxslt/libxslt_1.1.28-2.1ubuntu0.3/copyright

libglu1-mesa - 9.0.0-2.1

GNU General Public License, version 2, or any later version

GNU Library General Public License, version 2, or any later version
SGI FREE SOFTWARE LICENSE B (Version 2.0, Sept. 18, 2008)
Copyright (c) 1991-2000 Silicon Graphics, Inc. All Rights Reserved.
Copyright (c) 1995-1998 Brian Paul
Copyright (c) 2012 Timo Aaltonen

Download: http://archive.ubuntu.com/ubuntu/pool/main/libg/libglu/libglu_9.0.0.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libg/libglu/libglu_9.0.0-2.1/copyright with references to the full text of [GPL-2](#) and [LGPL-2](#)

lsf - 4.89+dfsg-0.1

SENDMAIL LICENSE

BSD 4-clause

Purdue license

GNU General Public License, version 2, or any later version

GNU Library General Public License, version 2, or any later version

Copyright (c) 1998 Sendmail, Inc. All rights reserved.

Copyright (c) Purdue Research Foundation, West Lafayette, Indiana 47907 2002 - 2011

Copyright (c) 1996, Dominik Kubla dominik@debian.org

Copyright (c) 1997, Michael Meskes meskes@debian.org

Copyright (c) 1998-2002, Jim Mintha jmintha@debian.org

Copyright (c) 2004-2009, Norbert Tretkowski nobse@debian.org

Copyright (c) 2012, Raoul Gunnar Borenus borenus@dfn.de

Copyright (c) 2012, Nicholas Bamber nicholas@periapt.co.uk

Copyright (c) 1993, Paul Kranenburg

Copyright (c) 2005-2007, Apple Computer, Inc. All rights reserved.

Copyright (c) 1983, 1993, The Regents of the University of California

Copyright (c) 1980, 1983, 1988, Regents of the University of California.

Copyright (c) 1983, 1988, 1993, Regents of the University of California.

Copyright (c) 2004, 2005, Fabian Frederick fabian.frederick@gmx.fr

Copyright (c) 1985, 1989-2000, Free Software Foundation, Inc.

Copyright (c) 1998, Sendmail, Inc. All rights reserved.

Copyright (c) 1997, Eric P. Allman. All rights reserved.

Copyright (c) 1988, 1993, The Regents of the University of California. All rights reserved.

Download: http://archive.ubuntu.com/ubuntu/pool/main/l/lsf/lsf_4.89+dfsg.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/l/lsf/lsf_4.89+dfsg-0.1/copyright

libcairo2 - 1.14.6-1

GNU Lesser General Public License, version 2.1 with additional copyright notice

Copyright (c) 1999 Tom Tromey

Copyright (c) 2002, 2003 University of Southern California, Information Sciences Institute (ISI)

Copyright (c) 2000, 2002, 2004, 2005 Keith Packard

Copyright (c) 2004 Calum Robinson

Copyright (c) 2004 Richard D. Worth

Copyright (c) 2004, 2005 Red Hat, Inc.

Copyright (c) 2004 David Reveman

Download: http://archive.ubuntu.com/ubuntu/pool/main/c/cairo/cairo_1.14.6.orig.tar.xz

License: http://changelogs.ubuntu.com/changelogs/pool/main/c/cairo/cairo_1.14.6-1/copyright with references to the full text of [LGPL-2.1](#)

locales - 2.23-0ubuntu11.3

Copyright (c) 1991-2015 Free Software Foundation, Inc.

Copyright (c) 1991 Regents of the University of California. All rights reserved.
Portions Copyright (c) 1993 by Digital Equipment Corporation.
Portions Copyright (c) 1996-1999 by Internet Software Consortium.
Copyright (c) 2010, Oracle America, Inc. Copyright (c) 1991,1990,1989 Carnegie Mellon University All Rights Reserved.

Copyright (c) 2000, Intel Corporation. All rights reserved.

Copyright (c) 1996 by Craig Metz, All Rights Reserved.

Copyright (c) 1992 Eric Young

Copyright (c) 2002, 2003, 2004, 2011 Simon Josefsson

Copyright (c) 1999, 2000 Tom Tromeey

Copyright (c) 2000 Red Hat, Inc.

Copyright (c) The Internet Society (2003). All Rights Reserved.

Copyright (c) 1998 WIDE Project. All rights reserved.

Copyright (c) 1995 by Tom Lord

Copyright (c) 1992, 1993, 1994, 1997 Henry Spencer. All rights reserved.

Copyright (c) 1997-2003 University of Cambridge

Copyright (c) 1993 by Sun Microsystems, Inc. All rights reserved.

(C) Copyright C E Chew

Copyright (c) 2001 by Stephen L. Moshier moshier@na-net.ornl.gov

Download: http://archive.ubuntu.com/ubuntu/pool/main/g/glibc/glibc_2.23.orig.tar.xz

License: http://changelogs.ubuntu.com/changelogs/pool/main/g/glibc/glibc_2.23-0ubuntu11.3/copyright with references to the full text of [GPL-2](#) and [LGPL-2.1](#)

ca-certificates - 20210119~16.04.1

GNU General Public License, version 2, or any later version

Mozilla Public License Version 2.0

Copyright (c) 2003 Fumitoshi UKAI ukai@debian.or.jp

Copyright (c) 2009 Philipp Kern pkern@debian.org

Copyright (c) 2011 Michael Shuler michael@pbandjelly.org

Copyright (c) Various Debian Contributors

Copyright (c) Mozilla Contributors

Download: http://archive.ubuntu.com/ubuntu/pool/main/c/ca-certificates/ca-certificates_20210119~16.04.1.tar.xz

License: http://changelogs.ubuntu.com/changelogs/pool/main/c/ca-certificates/ca-certificates_20210119~16.04.1/copyright with references to the full text of [GPL-2+](#)

PrizmDoc Application Services (PAS)

aws-sdk-js

Apache License Version 2.0

Copyright (c) 2012-2015 Amazon.com, Inc. or its affiliates. All Rights Reserved.

Download: <https://github.com/aws/aws-sdk-js>

License: <http://www.apache.org/licenses/LICENSE-2.0>

azure-storage

Apache License Version 2.0

Download: <https://github.com/Azure/azure-storage-node>

License: <https://github.com/Azure/azure-storage-node/blob/master/LICENSE.txt>

body-parser (<https://github.com/expressjs/body-parser>)

Copyright (c) 2014 Jonathan Ong <me@jongleberry.com>
Copyright (c) 2014-2015 Douglas Christopher Wilson <doug@somethingdoug.com>
Download: <https://github.com/expressjs/body-parser>
License: <https://github.com/expressjs/body-parser/blob/master/LICENSE>

caolan/async

The MIT License (MIT)
Copyright (c) 2010-2018 Caolan McMahon
Download: <https://github.com/caolan/async>
License: <https://github.com/caolan/async/blob/v1.4.2/LICENSE>

captains-log

The MIT License (MIT)
Copyright (c) 2012-present, Mike McNeil
Download: <https://www.npmjs.com/package/captains-log>
License: <https://github.com/balderdashy/sails/blob/master/LICENSE.md>

cors

The MIT License (MIT)
Copyright (c) 2013 Troy Goode troygoode@gmail.com
Download: <https://github.com/expressjs/cors>
License: <https://github.com/expressjs/cors/blob/master/LICENSE>

Duration.js

Copyright (c) 2013 Ilia Choly
Download: <https://github.com/icholy/Duration.js>
License: MIT <https://github.com/icholy/Duration.js/blob/master/LICENSE>

express (<http://expressjs.com/>)

Copyright (c) 2009-2014 TJ Holowaychuk <tj@vision-media.ca>
Copyright (c) 2013-2014 Roman Shtylman <shtylman+expressjs@gmail.com>
Copyright (c) 2014-2015 Douglas Christopher Wilson <doug@somethingdoug.com>
Download: <https://github.com/strongloop/express>
License: <https://github.com/strongloop/express/blob/master/LICENSE>

js-yaml (<https://github.com/nodeca/js-yaml>)

Copyright (c) 2011-2015 by Vitaly Puzrin
Download: <https://github.com/nodeca/js-yaml>
License: <https://github.com/nodeca/js-yaml/blob/master/LICENSE>

lodash

The MIT License (MIT)
Copyright (c) JS Foundation and other contributors <<https://js.foundation/>>
Copyright (c) OpenJS Foundation and other contributors <<https://openjsf.org/>>
Download: <https://github.com/lodash/lodash>
License: <https://github.com/lodash/lodash/blob/4.17/LICENSE>

lodash-node

Copyright jQuery Foundation and other contributors <<https://jquery.org/>>
Download: <https://www.npmjs.com/package/lodash-node>
License: <https://github.com/lodash-archive/lodash-node/blob/master/LICENSE>

mime-types

The MIT License (MIT)
Copyright (c) 2014 Jonathan Ong <me@jongleberry.com>
Copyright (c) 2015 Douglas Christopher Wilson <doug@somethingdoug.com>
Download: <https://www.npmjs.com/package/mime-types>
License: <https://github.com/jshttp/mime-types/blob/master/LICENSE>

mkdirp

The MIT License (MIT)
Copyright (c) 2010 James Halliday (mail@substack.net)
Download: <https://github.com/substack/node-mkdirp>
License: License: <https://github.com/substack/node-mkdirp/blob/master/LICENSE>

mv - 2.1.1

The MIT License (MIT)
Copyright (c) 2014 Andrew Kelley
Download: <https://github.com/andrewrk/node-mv>
License: <https://github.com/andrewrk/node-mv/blob/master/LICENSE>

node-bunyan

The MIT License (MIT)
Copyright (c) 2011-2012 Joyent Inc.
Download: <https://github.com/trentm/node-bunyan>
License: <https://github.com/trentm/node-bunyan/blob/master/LICENSE.txt>

node-mssql

Copyright (c) 2014 - 2016 Christopher Zotter
Download: <https://github.com/patriksimek/node-mssql>
License: <https://www.npmjs.com/package/node-mssql-connector#the-mit-license-mit>

node-mysql

The MIT License (MIT)
Copyright (c) 2012 Felix Geisendörfer (felix@debuggable.com) and contributors
Download: <https://github.com/felixge/node-mysql>
License: <install directory>/licenses/node-mysql/License

node-statsd

The MIT License (MIT)
Copyright 2011 Steve Ivy. All rights reserved.
Download: <https://www.npmjs.com/package/node-statsd>
License: <https://github.com/sivy/node-statsd/blob/master/LICENSE>

pm2 (<http://pm2.keymetrics.io/>)

Copyright (c) 2013-2015 Strzelewicz Alexandre

Download: <https://github.com/Unitech/PM2>
License: <https://github.com/Unitech/pm2/blob/master/GNU-AGPL-3.0.txt>

pump

The MIT License (MIT)
Copyright (c) 2014 Mathias Buus
Download: <https://github.com/mafintosh/pump>
License: <https://github.com/mafintosh/pump/blob/master/LICENSE>

request

Apache License Version 2.0
Version 2.0, January 2004
Download: <https://github.com/request/request>
License: <http://www.apache.org/licenses/>

rimraf

ISC license
Copyright (c) Isaac Z. Schlueter and Contributors
Download: <https://github.com/isaacs/rimraf>
License: <https://github.com/isaacs/rimraf/blob/master/LICENSE>

node-retry

Copyright: (c) 2011:
Tim Koschützki (tim@debuggable.com)
Felix Geisendörfer (felix@debuggable.com)
download: <https://github.com/tim-kos/node-retry>
license: MIT <https://github.com/tim-kos/node-retry/blob/master/License>

npm

The Artistic License 2.0
Copyright (c) npm, Inc. and Contributors
Download: <https://www.npmjs.com/package/npm>
License: <https://github.com/npm/cli/blob/latest/LICENSE>

sails-mssql

The MIT License (MIT)
Download: <https://github.com/intel/sails-mssql>
License: <https://opensource.org/licenses/MIT>

sails-mysql

The MIT License (MIT)
Copyright (c) 2016 Mike McNeil, Balderdash & contributors
Download: <https://github.com/balderdashy/sails-mysql>
License: <https://github.com/balderdashy/sails/blob/v0.12.2/LICENSE.md>

through2 (<https://github.com/rvagg/through2>)

Copyright (c) 2013, Rod Vagg (the "Original Author")
Download: <https://github.com/rvagg/through2>

License: <https://github.com/rvagg/through2>

underscore.string

The MIT License (MIT)

Copyright (c) 2011 Esa-Matti Suuronen esa-matti@suuronen.org

Download: <https://www.npmjs.com/package/underscore.string>

License: <https://github.com/esamattis/underscore.string>

waterline

Copyright (c) 2012-2016 Balderdash Design Co.

Download: <https://github.com/balderdashy/waterline>

License: MIT <https://github.com/balderdashy/waterline/blob/master/LICENSE.md>

waterline-cursor

The MIT License (MIT)

Copyright (c) 2012-present, Mike McNeil

Download: <https://www.npmjs.com/package/waterline-cursor>

License: <https://github.com/balderdashy/sails/blob/master/LICENSE.md>

yargs

Copyright (c) 2010 James Halliday (mail@substack.net)

Download: <https://github.com/bcoe/yargs>

License: MIT/X11 <https://github.com/bcoe/yargs/blob/master/LICENSE>

Viewer

jQuery (<http://jquery.com/>)

Copyright OpenJS Foundation and other contributors, <https://openjsf.org/>

Download: <http://jquery.com/download/>

Version: 3.6.0

License: <https://github.com/jquery/jquery/blob/master/LICENSE.txt>

jQuery Hotkeys

Copyright (c) 2010 by John Resig

Download: <https://plugins.jquery.com/hotkeys/>

License: <https://github.com/jeresig/jquery.hotkeys/blob/0.1.0/jquery.hotkeys.js#L4>

Normalize.css (<http://necolas.github.io/normalize.css/>)

Licensed under MIT license

Download: <https://github.com/necolas/normalize.css/>

Version: 3.0.0

License: <https://github.com/necolas/normalize.css/blob/3.0.0/LICENSE.md>

SVG - Material Design Icons

Licensed under the Pictogrammers Free License.

Download: <https://github.com/Templarian/MaterialDesign-SVG>

License: <https://github.com/Templarian/MaterialDesign-SVG/blob/v5.6.55/LICENSE>

The HTML5 Shiv (<https://code.google.com/p/html5shiv/>)

Dual licensed under the MIT or GPL Version 2 licenses

Download: <https://code.google.com/p/html5shiv/>

Version: 3.7.0

License: [https://github.com/aFarkas/html5shiv/blob/master/MIT and GPL2 licenses.md](https://github.com/aFarkas/html5shiv/blob/master/MIT%20and%20GPL2%20licenses.md)

Underscore (<http://underscorejs.org/>)

Copyright (c) 2009-2021 Jeremy Ashkenas, Julian Gonggrijp, and DocumentCloud and Investigative Reporters & Editors

Underscore may be freely distributed under the MIT license.

Download: <http://underscorejs.org/>

Version: 1.13.1

License: <https://github.com/jashkenas/underscore/blob/1.13.1/LICENSE>

Viewer Fonts

All fonts are licensed under the SIL Open Font License, version 1.1 - <http://scripts.sil.org/OFL>

Cedarville Cursive:

Copyright (c) 2010, Kimberly Geswein www.kimberlygeswein.com

Dancing Script:

Copyright (c) 2010, Pablo Impallari <https://github.com/impallari/DancingScript>; impallari@gmail.com

Copyright (c) 2010, Iginio Marini. www.ikern.com; mail@iginomarini.com

Fira Sans:

Copyright (c) 2012-2014, The Mozilla Foundation and Telefonica S.A.

Grand Hotel:

Copyright (c) 2012, by Brian J. Bonislawsky and Jim Lyles DBA Astigmatic (AOETI) (astigma@astigmatic.com)

Great Vibes:

Copyright (c) 2012, TypeSETit, LLC (typesetit@att.net)

Italianno:

Copyright (c) 2011, TypeSETit, LLC (typesetit@att.net)

La Belle Aurore:

Copyright (c) 2010, Kimberly Geswein (www.kimberlygeswein.com)

Pacifico:

Copyright (c) 2011, Vernon Adams (vern@newtypography.co.uk)

PT Mono:

Copyright (c) 2011, ParaType Ltd. (<http://www.paratype.com/public>)

PT Serif:

Copyright (c) 2010, ParaType Ltd. All rights reserved.

Sacramento:

Copyright (c) 2012, Brian J. Bonislawsky DBA Astigmatic (AOETI) (astigma@astigmatic.com)

@prizmdoc/viewer-core

D3.js

Copyright (c) 2010-2014, Michael Bostock

Download: <https://github.com/d3/d3/blob/2a7d873b3fad4c808b2724db9b0663aa476583b/src/svg/line.js>

License: <https://github.com/d3/d3/blob/2a7d873b3fad4c808b2724db9b0663aa476583b/LICENSE>

David Chamber's base64 encoder and decoder

Copyright (c) 2011..2012 David Chambers dc@hashify.me

Download:

<https://github.com/davidchambers/Base64.js/blob/8fe0eda2391db0e4a867753e0dd568e8fd0cfc31/base64.js>

License:

<https://github.com/davidchambers/Base64.js/blob/8fe0eda2391db0e4a867753e0dd568e8fd0cfc31/LICENSE>

better-random-numbers-for-javascript

Copyright (c) 2010 by Johannes Baagøe baagoe@baagoe.org

Download: <https://github.com/nquinlan/better-random-numbers-for-javascript-mirror/blob/master/support/js/Alea.js>

License: <https://github.com/nquinlan/better-random-numbers-for-javascript-mirror/blob/master/LICENSE.md>

PrizmDoc Viewer Eval Docker Image

ca-certificates - 20210119~16.04.1

GNU General Public License, version 2, or any later version

Mozilla Public License Version 2.0

Copyright (c) 2003 Fumitoshi UKAI ukai@debian.or.jp

Copyright (c) 2009 Philipp Kern pkern@debian.org

Copyright (c) 2011 Michael Shuler michael@pbandjelly.org

Copyright (c) Various Debian Contributors

Copyright (c) Mozilla Contributors

Download: http://archive.ubuntu.com/ubuntu/pool/main/c/ca-certificates/ca-certificates_20210119~16.04.1.tar.xz

License: http://changelogs.ubuntu.com/changelogs/pool/main/c/ca-certificates/ca-certificates_20210119~16.04.1/copyright with references to the full text of [GPL-2+](#)

consolidate

The MIT License (MIT)

Copyright (c) 2011-2016 TJ Holowaychuk tj@vision-media.ca

Download: <https://www.npmjs.com/package/consolidate>

License: <https://github.com/tj/consolidate.js#license>

curl - 7.47.0-1ubuntu2.19

curl License

ISC License

BSD 3-Clause

BSD 4-Clause

public-domain

Copyright (c) 1996-2015, Daniel Stenberg daniel@haxx.se

Copyright (c) 2010, DirecTV

2010-2015, Daniel Stenberg daniel@haxx.se

Copyright (c) 2012-2014, Nick Zitzmann nickzman@gmail.com 2012-2015, Daniel Stenberg daniel@haxx.se

Copyright (c) 2010, Howard Chu hyc@highlandsun.com

Copyright (c) 2012-2014, Marc Hoersken info@marc-hoersken.de

2012, Mark Salisbury mark.salisbury@hp.com

2012-2015, Daniel Stenberg daniel@haxx.se

Copyright (c) 1996-2001 Internet Software Consortium

Copyright (c) 2004-2015 Daniel Stenberg

1995-1999 Kungliga Tekniska Högskolan

Copyright (c) 2001, Solar Designer solar@openwall.com

Copyright (c) 2011-2015, Daniel Stenberg daniel@haxx.se

2010, Howard Chu hyc@openldap.org

Copyright (c) 2010-2011, Hoi-Ho Chan hoiho.chan@gmail.com

2012-2015, Daniel Stenberg daniel@haxx.se

Copyright (c) 2009, 2011, Markus Moeller, markus_moeller@compuserve.com

2012-2015, Daniel Stenberg, daniel@haxx.se

Copyright (c) 2000-2009, EdelWeb for EdelKey and OpenEvidence

Copyright (c) 1983 Regents of the University of California

Copyright (c) 2010, Mandy Wu mandy.wu@intel.com

2011-2013, Daniel Stenberg daniel@haxx.se

Copyright (c) 2003, Simtec Electronics

Copyright (c) 2011, Jim Hollinger

Copyright (c) 2003, The OpenEvidence Project

Copyright (c) 2001, Eric Lavigne

Copyright (c) 2000-2010, Domenico Andreoli cavok@debian.org

2010-2011, Ramakrishnan Muthukrishnan rkrishnan@debian.org

2011, Alessandro Ghedini ghedo@debian.org

Download: http://archive.ubuntu.com/ubuntu/pool/main/c/curl/curl_7.47.0.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/c/curl/curl_7.47.0-1ubuntu2.19/copyright

express

The MIT License (MIT)

Copyright (c) 2009-2014 TJ Holowaychuk tj@vision-media.ca

Copyright (c) 2013-2014 Roman Shtylman shtylman+expressjs@gmail.com

Copyright (c) 2014-2015 Douglas Christopher Wilson doug@somethingdoug.com

Download: <https://www.npmjs.com/package/express>

License: <https://cdn.jsdelivr.net/npm/express@4.16.4/LICENSE>

fontconfig - 2.11.94-0ubuntu1.1

Copyright (c) 2001, 2003 Keith Packard

Download: http://archive.ubuntu.com/ubuntu/pool/main/f/fontconfig/fontconfig_2.11.94.orig.tar.bz2

License: http://changelogs.ubuntu.com/changelogs/pool/main/f/fontconfig/fontconfig_2.11.94-

[0ubuntu1.1/copyright](#)

handlebars

The MIT License (MIT)

Copyright (C) 2011-2017 by Yehuda Katz

Download: <https://www.npmjs.com/package/handlebars>

License: <https://cdn.jsdelivr.net/npm/handlebars@4.7.6/LICENSE>

http-errors

The MIT License (MIT)

Copyright (c) 2014 Jonathan Ong me@jongleberry.com

Copyright (c) 2016 Douglas Christopher Wilson doug@somethingdoug.com

Download: <https://www.npmjs.com/package/http-errors>

License: <https://cdn.jsdelivr.net/npm/http-errors@1.6.3/LICENSE>

http-proxy-middleware

The MIT License (MIT)

Copyright (c) 2015 Steven Chim

Download: <https://www.npmjs.com/package/http-proxy-middleware>

License: <https://cdn.jsdelivr.net/npm/http-proxy-middleware@0.19.1/LICENSE>

libcairo2 - 1.14.6-1

GNU Lesser General Public License, version 2.1 with additional copyright notice

Copyright (c) 1999 Tom Tromey

Copyright (c) 2002, 2003 University of Southern California, Information Sciences Institute (ISI)

Copyright (c) 2000, 2002, 2004, 2005 Keith Packard

Copyright (c) 2004 Calum Robinson

Copyright (c) 2004 Richard D. Worth

Copyright (c) 2004, 2005 Red Hat, Inc.

Copyright (c) 2004 David Reveman

Download: http://archive.ubuntu.com/ubuntu/pool/main/c/cairo/cairo_1.14.6.orig.tar.xz

License: http://changelogs.ubuntu.com/changelogs/pool/main/c/cairo/cairo_1.14.6-1/copyright with references to the full text of [LGPL-2.1](#)

libcups2 - 2.1.3-4ubuntu0.11

GNU General Public License, version 2 with Apple Operating System Development License Exception

GNU Library General Public License, version 2 with Apple Operating System Development License Exception

Zlib license

BSD 2-clause

Copyright (c) 2007-2015, Apple Inc.

Copyright (c) 2007-2013, Apple Inc.

Copyright (c) 2005, Easy Software Products

Copyright (c) 1999, Aladdin Enterprises.

Copyright (c) 2011, Red Hat, Inc

Copyright (c) 2007-2014, Apple Inc

Copyright (c) 1997-2007, Easy Software Products

Download: http://archive.ubuntu.com/ubuntu/pool/main/c/cups/cups_2.1.3.orig.tar.bz2

License: http://changelogs.ubuntu.com/changelogs/pool/main/c/cups/cups_2.1.3-4ubuntu0.11/copyright with references to the full text of [GPL-2.0](#) and [LGPL-2.0](#)

libdbus-glib-1-2 - 0.106-1

GNU General Public License, version 2 or any later version

Expat license

Copyright (c) 2002-2010 Red Hat, Inc

Copyright (c) 2002-2003 CodeFactory AB

Copyright (c) 2004 Ximian, Inc

Copyright (c) 2005-2011 Nokia Corporation

Copyright (c) 2006 Steve Frécinaux

Copyright (c) 2007 Codethink Ltd

Copyright (c) 2009-2014 Collabora Ltd

Copyright (c) 2013 Intel Corporation

Copyright (c) 2008 David Zeuthen

Copyright (c) 2009 Collabora Ltd

Copyright (c) 2009-2011 Nokia Corporation

Copyright (c) 2011 Nokia Corporation

Copyright (c) 2013 Collabora Ltd

Download: http://archive.ubuntu.com/ubuntu/pool/main/d/dbus-glib/dbus-glib_0.106.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/d/dbus-glib/dbus-glib_0.106-1/copyright with references to the full text of [GPL-2+](#)

libexif12 - 0.6.21-2ubuntu0.6

GNU General Public License, version 2 or any later version

GNU Lesser General Public License, version 2.1 or any later version

BSD 2-clause

Copyright (c) 2001-2009, Lutz Müller urc8@rz.uni-karlsruhe.de

Copyright (c) 2004-2009, Jan Patera patera@users.sourceforge.net

Copyright (c) 2004, Joerg Hoh joerg@devone.org

Copyright (c) 2005-2006, Hubert Figuiere hub@figuiere.net

Copyright (c) 2002-2005, Hans Ulrich Niedermann gp@n-dimensional.de

Copyright (c) 2007-2010, Dan Fandrich dan@cone harvesters.com

Copyright (c) 2002, Fredrik fredrik@krixor.xy.org

Copyright (c) 2002, Javier Achirica achirica@ttd.net

Copyright (c) 2002, Semyon Sosin sem@best.com

Copyright (c) 2002, Guido Ostkamp guido.ostkamp@t-online.de

Copyright (c) 2002, Takuro Ashie makeinu@users.sourceforge.net

Copyright (c) 2002, Jason Sodergren jason@taiga.com

Copyright (c) 2002, Renchi Raju renchi@pooh.tam.uiuc.edu

Copyright (c) 2003, Torgeir Hansen torgeir@trenger.ro

Copyright (c) 2003, Roberto Costa roberto.costa@ensta.org

Copyright (c) 2004, Angela Wrobel

Copyright (c) 2002, Basil Dias basil.dias@wipro.com

Copyright (c) 2003, Ralph Heidelberg RHeidelberg@Pinnaclesys.com

Copyright (c) 2002, Javier Achirica achirica@ttd.net

Copyright (c) 2003, Gernot Jander gernot@bigpond.com

Copyright (c) 2004, Antonio Scuri scuri@tecgraf.puc-rio.br

Copyright (c) 2002, Mark Pulford mark@kyne.com.au

Copyright (c) 2002, Semyon Sosin sem@best.com

Copyright (c) 2002, Marcus Meissner marcus@jet.franken.de

Copyright (c) 2003, Jens Finke jens@triq.net

Copyright (c) 2007, Meder Kydyraliev

Copyright (c) 2008, Mika Raento mikie@google.com

Copyright (c) 2003, Peter Bieringer pb@bieringer.de

Copyright (c) 2004, Antonio Scuri scuri@tecgraf.puc-rio.br

Copyright (c) 2008-2010, Erwin Poeze

Copyright (c) 2008-2010, Marcus Meissner
Copyright (c) 2008-2010, Jakub Bogusz
Copyright (c) 2008, Ivan Masár
Copyright (c) 2008-2010, Clytie Siddall
Copyright (c) 2009, Daniel Nylander
Copyright (c) 2009, Jan Patera
Copyright (c) 2009-2010, Marko Uskokovic
Copyright (c) 2009-2010, Tao Wei weitao1979@gmail.com
Copyright (c) 2009-2010, Vilson Gjerci vilsongjeci@gmail.com
Copyright (c) 2009, nglNx
Copyright (c) 2010, Launchpad Translators
Copyright (c) 2009, Shushi Kurose
Copyright (c) 2009-2010, Tadashi Jokagi
Copyright (c) 2009-2010, Sergio Zanchetta
Copyright (c) 2009, Bruce Cowan
Copyright (c) 2009-2010, Iryna Nikanchuk defragbrain@gmail.com
Copyright (c) 2009-2010, Joe Hansen
Copyright (c) 2010, Enes Ateş e.n.3.s@hotmail.com
Copyright (c) 2010, Launchpad Translators
Copyright (c) 2010, Robert Readman
Copyright (c) 2010, André Gondim andregondim@ubuntu.com
Copyright (c) 2010, Alexey Ivanov alexey.ivanov@gmail.com
Copyright (c) 2009, André Gondi
Copyright (c) 2002-2004, Christophe Barbe christophe@debian.org
Copyright (c) 2004-2007, Frederic Peters fpeters@debian.org
Copyright (c) 2009-2011, Emmanuel Bouthenot kolter@debian.org
Download: http://archive.ubuntu.com/ubuntu/pool/main/libe/libexif/libexif_0.6.21.orig.tar.gz
License: http://changelogs.ubuntu.com/changelogs/pool/main/libe/libexif/libexif_0.6.21-2ubuntu0.6/copyright
with references to the full text of [GPL-2+](#) and [LGPL-2.1+](#)

libexpat1 - 2.1.0-7ubuntu0.16.04.5

The MIT License (MIT)
Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd and Clark Cooper
Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006 Expat maintainers
Download: http://archive.ubuntu.com/ubuntu/pool/main/e/expat/expat_2.1.0.orig.tar.gz
License: http://changelogs.ubuntu.com/changelogs/pool/main/e/expat/expat_2.1.0-7ubuntu0.16.04.5/copyright

libfreetype6 - 2.6.1-0.1ubuntu2.4

GNU General Public License, version 2 or any later version
Zlib license
BSD 2-clause
BSD 3-clause
The Catharon Open Source LICENSE
The OpenGroup BSD-like license Copyright (c) 1996-2012 David Turner, Robert Wilhelm, and Werner Lemberg
Copyright (c) 1996-2009 Just van Rossum
Copyright (c) 2002-2012 Roberto Alameda
Copyright (c) 2003 Huw D M Davies for Codeweavers
Copyright (c) 2003-2012 Masatake YAMATO, Redhat K.K.
Copyright (c) 2004-2012 Albert Chin-A-Young
Copyright (c) 2004-2012 Suzuki Toshiya
Copyright (c) 2007 Dmitry Timoshkov for Codeweavers
Copyright (c) 2007-2011 Rahul Bhalariao rahul.bhalerao@redhat.com
Copyright (c) 2007-2012 Derek Clegg, Michael Toftdal

Copyright (c) 2009-2011 Oran Agra, Mickey Gabel

Copyright (c) 2010, 2012 Joel Klinghed

Copyright (c) 1996-2012 Christoph Lameter clameter@waterf.org, Anthony Fok foka@debian.org, Steve Langasek vorlon@debian.org, et al.

Copyright (c) 2001, 2002 Catharon Productions Inc.

Copyright (c) 1995-2002 Jean-loup Gailly and Mark Adler

Copyright (c) 2002-2006, 2009-2012 David Turner, Robert Wilhelm, Werner Lemberg

Copyright (c) 2005-2008 George Williams

Copyright (c) 2000-2012 Francesco Zappa Nardelli francesco.zappa.nardelli@ens.fr

Copyright (c) 2000 Computing Research Labs, New Mexico State University

Copyright (c) 1990, 1994, 1998 The Open Group

Download: http://archive.ubuntu.com/ubuntu/pool/main/f/freetype/freetype_2.6.1.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/f/freetype/freetype_2.6.1-0.1ubuntu2.4/copyright with references to the full text of [GPL-2+](#)

libglu1-mesa - 9.0.0-2.1

GNU General Public License, version 2, or any later version

GNU Library General Public License, version 2, or any later version

SGI FREE SOFTWARE LICENSE B (Version 2.0, Sept. 18, 2008)

Copyright (c) 1991-2000 Silicon Graphics, Inc. All Rights Reserved.

Copyright (c) 1995-1998 Brian Paul

Copyright (c) 2012 Timo Aaltonen

Download: http://archive.ubuntu.com/ubuntu/pool/main/libg/libglu/libglu_9.0.0.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libg/libglu/libglu_9.0.0-2.1/copyright with references to the full text of [GPL-2](#) and [LGPL-2](#)

libjpeg62 - 1:6b2-2

Copyright (C) 1991-1998, Thomas G. Lane.

The Graphics Interchange Format(c) is the Copyright property of CompuServe Incorporated.

GIF(sm) is a Service Mark property of CompuServe Incorporated.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libj/libjpeg6b/libjpeg6b_6b2.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/universe/libj/libjpeg6b/libjpeg6b_6b2-2/copyright

libpixmap-1-0 - 0.33.6-1

The MIT License (MIT)

Copyright (c) 1987, 1988, 1989, 1998 The Open Group

Copyright (c) 1987, 1988, 1989 Digital Equipment Corporation

Copyright (c) 1999, 2004, 2008 Keith Packard

Copyright (c) 2000 SuSE, Inc.

Copyright (c) 2000 Keith Packard, member of The XFree86 Project, Inc.

Copyright (c) 2004, 2005, 2007, 2008, 2009, 2010 Red Hat, Inc.

Copyright (c) 2004 Nicholas Miell

Copyright (c) 2005 Lars Knoll & Zack Rusin, Trolltech

Copyright (c) 2005 Trolltech AS

Copyright (c) 2007 Luca Barbato

Copyright (c) 2008 Aaron Plattner, NVIDIA Corporation

Copyright (c) 2008 Rodrigo Kumpera

Copyright (c) 2008 Andr   Tupinamb   Copyright (c) 2008 Mozilla Corporation

Copyright (c) 2008 Frederic Plourde

Copyright (c) 2009, Oracle and/or its affiliates. All rights reserved.

Copyright (c) 2009, 2010 Nokia Corporation

Download: http://archive.ubuntu.com/ubuntu/pool/main/p/pixmap/pixmap_0.33.6.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/p/pixman/pixman_0.33.6-1/copyright

libssl1.0.0 - 1.0.2g-1ubuntu4.19

BSD License

Copyright (c) 1998-2004 The OpenSSL Project

Copyright (c) 1995-1998 Eric A. Young, Tim J. Hudson

Copyright (c) 1998-2004 The OpenSSL Project. All rights reserved.

Copyright (C) 1995-1998 Eric Young (ey@cryptsoft.com). All rights reserved.

Download: http://archive.ubuntu.com/ubuntu/pool/main/o/openssl/openssl_1.0.2g.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/o/openssl/openssl_1.0.2g-1ubuntu4.19/copyright

libx11-6 - 2:1.6.3-1ubuntu2.2

The MIT License (MIT)

Copyright (c) 2003-2006,2008 Jamey Sharp, Josh Triplett

Copyright (c) 2009 Red Hat, Inc.

Copyright (c) 1990-1992,1999,2000,2004,2009,2010 Oracle and/or its affiliates. All rights reserved.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libx11/libx11_1.6.3.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libx11/libx11_1.6.3-1ubuntu2.2/copyright

libxau6 - 1:1.0.8-1

Copyright (c) 1988, 1998 The Open Group

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libxau/libxau_1.0.8.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libxau/libxau_1.0.8-1/copyright

libxext6 - 2:1.3.3-1

Copyright (c) 1986, 1987, 1988, 1989, 1994, 1998 The Open Group

Copyright (c) 1996 Digital Equipment Corporation, Maynard, Massachusetts.

Copyright (c) 1997 by Silicon Graphics Computer Systems, Inc.

Copyright (c) 1992 Network Computing Devices

Copyright (c) 1991,1993 by Digital Equipment Corporation, Maynard, Massachusetts, and Olivetti Research Limited, Cambridge, England.

Copyright (c) 1986, 1987, 1988 by Hewlett-Packard Corporation

Copyright (c) 1994, 1995 Hewlett-Packard Company

Copyright (c) Digital Equipment Corporation, 1996

Copyright (c) 1999, 2005, 2006, Oracle and/or its affiliates. All rights reserved.

Copyright (c) 1989 X Consortium, Inc. and Digital Equipment Corporation.

Copyright (c) 1992 X Consortium, Inc. and Intergraph Corporation.

Copyright (c) 1993 X Consortium, Inc. and Silicon Graphics, Inc.

Copyright (c) 1994, 1995 X Consortium, Inc. and Hewlett-Packard Company.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libxext/libxext_1.3.3.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libxext/libxext_1.3.3-1/copyright

libxinerama1 - 2:1.1.3-1

Copyright (c) 2007, Oracle and/or its affiliates. All rights reserved.

Copyright (c) 2003 The Open Group

Copyright (c) 1991, 1997 Digital Equipment Corporation, Maynard, Massachusetts.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libxinerama/libxinerama_1.1.3.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libxinerama/libxinerama_1.1.3-1/copyright

libxrender1 - 1:0.9.9-0ubuntu1

Copyright (c) 2001,2003 Keith Packard

Copyright (c) 2000 SuSE, Inc.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libxrender/libxrender_0.9.9.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libxrender/libxrender_0.9.9-0ubuntu1/copyright

libxslt1.1 - 1.1.28-2.1ubuntu0.3

Copyright (c) 2001-2002 Daniel Veillard. All Rights Reserved.

Copyright (c) 2001-2002 Thomas Broyer, Charlie Bozeman and Daniel Veillard. All Rights Reserved.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libxslt/libxslt_1.1.28.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libxslt/libxslt_1.1.28-2.1ubuntu0.3/copyright

libxtst6 - 2:1.2.2-1

Copyright (c) 1990, 1991 by UniSoft Group Limited

Copyright (c) 1992, 1993, 1995, 1998 The Open Group

Copyright (c) 1995 Network Computing Devices

Copyright (c) 2005 Red Hat, Inc.

Copyright (c) 1992 by UniSoft Group Ltd.

Copyright (c) 1992, 1994, 1995 X Consortium

Copyright (c) 1994 Network Computing Devices, Inc.

Download: http://archive.ubuntu.com/ubuntu/pool/main/libx/libxtst/libxtst_1.2.2.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/libx/libxtst/libxtst_1.2.2-1/copyright

locales - 2.23-0ubuntu11.3

Copyright (c) 1991-2015 Free Software Foundation, Inc.

Copyright (c) 1991 Regents of the University of California. All rights reserved.

Portions Copyright (c) 1993 by Digital Equipment Corporation.

Portions Copyright (c) 1996-1999 by Internet Software Consortium.

Copyright (c) 2010, Oracle America, Inc. Copyright (c) 1991,1990,1989 Carnegie Mellon University All Rights Reserved.

Copyright (c) 2000, Intel Corporation. All rights reserved.

Copyright (c) 1996 by Craig Metz, All Rights Reserved.

Copyright (c) 1992 Eric Young

Copyright (c) 2002, 2003, 2004, 2011 Simon Josefsson

Copyright (c) 1999, 2000 Tom Trome

Copyright (c) 2000 Red Hat, Inc.

Copyright (c) The Internet Society (2003). All Rights Reserved.

Copyright (c) 1998 WIDE Project. All rights reserved.

Copyright (c) 1995 by Tom Lord

Copyright (c) 1992, 1993, 1994, 1997 Henry Spencer. All rights reserved.

Copyright (c) 1997-2003 University of Cambridge

Copyright (c) 1993 by Sun Microsystems, Inc. All rights reserved.

(C) Copyright C E Chew

Copyright (c) 2001 by Stephen L. Moshier moshier@na-net.ornl.gov

Download: <https://packages.ubuntu.com/source/xenial/glibc>

License: http://changelogs.ubuntu.com/changelogs/pool/main/g/glibc/glibc_2.23-0ubuntu11.3/copyright with references to the full text of [GPL-2](#) and [LGPL-2.1](#)

isof - 4.89+dfsg-0.1

SENDMAIL LICENSE

BSD 4-clause

Purdue license

GNU General Public License, version 2, or any later version

GNU Library General Public License, version 2, or any later version

Copyright (c) 1998 Sendmail, Inc. All rights reserved.

Copyright (c) Purdue Research Foundation, West Lafayette, Indiana 47907 2002 - 2011

Copyright (c) 1996, Dominik Kubla dominik@debian.org

Copyright (c) 1997, Michael Meskes meskes@debian.org

Copyright (c) 1998-2002, Jim Mintha jmintha@debian.org

Copyright (c) 2004-2009, Norbert Tretkowski nobse@debian.org

Copyright (c) 2012, Raoul Gunnar Borenus borenus@dfn.de

Copyright (c) 2012, Nicholas Bamber nicholas@periapt.co.uk

Copyright (c) 1993, Paul Kranenburg

Copyright (c) 2005-2007, Apple Computer, Inc. All rights reserved.

Copyright (c) 1983, 1993, The Regents of the University of California

Copyright (c) 1980, 1983, 1988, Regents of the University of California.

Copyright (c) 1983, 1988, 1993, Regents of the University of California.

Copyright (c) 2004, 2005, Fabian Frederick fabian.frederick@gmx.fr

Copyright (c) 1985, 1989-2000, Free Software Foundation, Inc.

Copyright (c) 1998, Sendmail, Inc. All rights reserved.

Copyright (c) 1997, Eric P. Allman. All rights reserved.

Copyright (c) 1988, 1993, The Regents of the University of California. All rights reserved.

Download: http://archive.ubuntu.com/ubuntu/pool/main/l/lsof/lsof_4.89+dfsg.orig.tar.gz

License: http://changelogs.ubuntu.com/changelogs/pool/main/l/lsof/lsof_4.89+dfsg-0.1/copyright

node-fetch

The MIT License (MIT)

Copyright (c) 2016 David Frank

Download: <https://www.npmjs.com/package/node-fetch>

License: <https://cdn.jsdelivr.net/npm/node-fetch@2.6.0/LICENSE.md>

node-rsa

The MIT License (MIT)

Copyright (c) 2014 rzcoder

Copyright (c) 2003-2005 Tom Wu

Download: <https://www.npmjs.com/package/node-rsa>

License: <https://github.com/rzcoder/node-rsa/blob/1.1.1/README.md>

Node.js v12.22.1

Copyright Node.js contributors. All rights reserved.

Download: <http://nodejs.org/dist/v12.22.1/>

License: <https://github.com/nodejs/node/blob/v12.22.1/LICENSE>

Legacy Samples

ASP.NET MVC

Copyright (c) Microsoft Corporation

Download: <https://www.nuget.org/packages/Microsoft.AspNet.Mvc/5.2.7/>

License: https://www.microsoft.com/web/webpi/eula/net_library_eula_ENU.htm

ASP.NET Razor

Copyright (c) Microsoft Corporation

Download: <https://www.nuget.org/packages/Microsoft.AspNet.Razor/3.2.2>

License: https://www.microsoft.com/web/webpi/eula/net_library_eula_ENU.htm

ASP.NET Web Pages

Copyright (c) Microsoft Corporation

Download: <https://www.nuget.org/packages/Microsoft.AspNet.WebPages/3.2.2>

License: https://www.microsoft.com/web/webpi/eula/net_library_eula_ENU.htm

Font Awesome Free

Download: <https://github.com/FortAwesome/Font-Awesome/>

License: <https://scripts.sil.org/OFL>

dropzone.js

The MIT License (MIT)

Copyright (c) 2012 Matias Meno

Download: <https://github.com/enyo/dropzone>

License: <https://github.com/enyo/dropzone/blob/master/LICENSE>

infinity.js (<http://airbnb.io/infinity/>)

Copyright (c) 2012 Airbnb

Download: <https://github.com/airbnb/infinity>

License: <https://github.com/airbnb/infinity/blob/master/LICENSE>

Microsoft.Web.Infrastructure

Download: <https://www.nuget.org/packages/Microsoft.Web.Infrastructure/1.0.0/>

License: <https://www.microsoft.com/web/webpi/eula/aspnetmvc3update-eula.htm>

Newtonsoft.Json

Copyright (c) 2007 James Newton-King

Download: <http://www.newtonsoft.com/json>

License: <https://github.com/JamesNK/Newtonsoft.Json/blob/7.0.1/LICENSE.md>

pikaday

Copyright (c) 2014 David Bushell

Download: <https://github.com/Pikaday/Pikaday/tree/1.3.2>

License: <https://github.com/Pikaday/Pikaday/blob/1.3.2/LICENSE>

jQuery (<http://jquery.com/>)

Copyright OpenJS Foundation and other contributors, <https://openjsf.org/>

Download: <http://jquery.com/download/>

Version: 3.6.0

License: <https://github.com/jquery/jquery/blob/master/LICENSE.txt>

jQuery Hotkeys

Copyright (c) 2010 by John Resig

Download: <https://plugins.jquery.com/hotkeys/>

License: <https://github.com/jeresig/jquery.hotkeys/blob/0.1.0/jquery.hotkeys.js#L4>

JSP Sample

Json library: Json is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object.

Download: <https://code.google.com/p/google-son/>

Version: 2.2.4

License: <http://www.apache.org/licenses/LICENSE-2.0>

SVG - Material Design Icons

Licensed under the Pictogrammers Free License.

Download: <https://github.com/Templarian/MaterialDesign-SVG>

License: <https://github.com/Templarian/MaterialDesign-SVG/blob/v5.6.55/LICENSE>

The HTML5 Shiv (<https://code.google.com/p/html5shiv/>)

Dual licensed under the MIT or GPL Version 2 licenses

Download: <https://code.google.com/p/html5shiv/>

Version: 3.7.0

License: [https://github.com/aFarkas/html5shiv/blob/master/MIT and GPL2 licenses.md](https://github.com/aFarkas/html5shiv/blob/master/MIT%20and%20GPL2%20licenses.md)

Underscore (<http://underscorejs.org/>)

Copyright (c) 2009-2021 Jeremy Ashkenas, Julian Gonggrijp, and DocumentCloud and Investigative Reporters & Editors

Underscore may be freely distributed under the MIT license.

Download: <http://underscorejs.org/>

Version: 1.13.1

License: <https://github.com/jashkenas/underscore/blob/1.13.1/LICENSE>

Windows Installer XML (Wi) toolset

Copyright (c) 2004, Outer Curve Foundation

Download: <https://wix.codeplex.com/releases/view/99514>

License: <http://opensource.org/licenses/ms-url>

Release Notes

This section contains information on new features, improvements, fixes, and known issues for each release:

- [Release Notes v13.17](#)
- [Release Notes v13.16](#)
- [Release Notes v13.15](#)
- [Release Notes v13.14](#)
- [Release Notes v13.13](#)
- [Release Notes v13.12](#)
- [Release Notes v13.11](#)
- [Release Notes v13.10](#)
- [Release Notes v13.9](#)
- [Release Notes v13.8](#)
- [Release Notes v13.7](#)
- [Release Notes v13.6](#)
- [Release Notes v13.5](#)
- [Release Notes v13.4](#)
- [Release Notes v13.3](#)
- [Release Notes v13.2](#)
- [Release Notes v13.1](#)
- [Release Notes v13.0](#)
- [Known Issues](#)

Release Notes v13.17

PrizmDoc Viewer v13.17 introduces the following:

- [New Features & Improvements](#)
- [Product Updates & Fixes](#)

13.17 New Features and Improvements

- **Content Conversion Service (CCS) now allows you to control the conversion of MS Word documents with tracked changes when using the MSO rendering engine.** If you are using [PrizmDoc Server with Microsoft Office](#), you can now use the Content Conversion Service to convert Microsoft Word documents with accepted or rejected markup changes to view those documents in their final form. See the [API guide](#) for more information.
- **Content Conversion Service (CCS) now allows you to control the conversion of MS PowerPoint documents with speaker notes when using the MSO rendering engine.** If you are using [PrizmDoc Server with Microsoft Office](#), you can now use the Content Conversion Service to convert Microsoft PowerPoint documents with slides only (the default) or with included speaker notes to view those documents in their final form. See the [API guide](#) for more information.
- **Improved support of signed PDF documents.** Improved PDF rendering service to better support signatures in encrypted PDF files.
- **CentOS 6 and Red Hat Enterprise Linux 6 are no longer supported.** CentOS 6 support was deprecated in PrizmDoc Viewer v13.14. Red Hat Enterprise Linux 6 support was deprecated in PrizmDoc Viewer v13.16. As of PrizmDoc Viewer v13.17, support for CentOS 6 and Red Hat Enterprise Linux 6 has been dropped. Please consider using PrizmDoc Viewer Docker images ([PrizmDoc Server](#) or [PrizmDoc Application Services](#)) or upgrading your system.
- **Ubuntu 16.04 is no longer supported.** Ubuntu 16.04 support was deprecated in PrizmDoc Viewer v13.16. As of PrizmDoc Viewer v13.17, support for Ubuntu 16.04 has been dropped. Please consider using PrizmDoc

Viewer Docker images ([PrizmDoc Server](#) or [PrizmDoc Application Services](#)) or upgrading your system.

- While we currently continue to offer and support traditional Linux packages for direct installation on a Linux host, these have largely become obsolete now that Docker deployment is an option. **We have announced deprecation of our traditional Linux install packages with the release of PrizmDoc Viewer v13.17**, and, in a future product release, we intend to only offer our [Docker-based deployment option](#).
- **PrizmDoc Viewer [Microsoft Office conversion add-on](#) option is now compatible with Microsoft Office 2019 for PrizmDoc Server running on Windows 2019 platform.** Please refer to the [Office Issues](#) section to review and understand Microsoft's known performance issue in Excel 2019 when considering migrating your production environment to Microsoft Office 2019.

13.17 Product Updates and Fixes

Viewing

- Addressed an issue where selecting a search result of another page did not bring the search result into view while in single page view mode.
- Addressed an issue where the redaction search results were not correctly updated when performing a search that excluded the document.
- Addressed an issue in the Viewer where the wrong search result was selected when selecting the previous or next search result. This occurred while a search with multiple pages of results was in progress.
- Addressed an issue in the Viewer where certain regular expression searches failed or caused the browser to hang. The Viewer no longer modifies regular expression searches to handle finding phrases that span multiple lines or contain variations of single or double quote characters. Lines of text in the Viewer typically end with a space followed by a \n character, so to include phrases that span multiple lines in your regular expression search results, you will need to provide a regular expression that accounts for words separated by either a space or a space followed by a \n character.
- Addressed an issue in the Viewer where searching for terms that include single or double quote characters did not return the correct results.
- Addressed an issue in the Viewer where using the search filter panel to exclude a search term did not unhighlight the search result in a previously selected comment.
- Addressed an issue when MSG document that contains an attachment with meeting info in it fails to load.
- Addressed an issue where the Viewer was preserving search highlights on comments in the saved markup.
- Addressed an issue where EML document with HTML body charset specified as empty string fails to load.

Redaction

- Addressed an issue with the PrizmDoc Server [markupBurners REST API](#) where it was unable to apply markup definitions to signed PDF files.

Fidelity

- Addressed an issue where embedded cyrillic fonts in PDF documents were not being rendered.
- Addressed a limitation where JBIG2 streams in some specific PDF documents were not being rendered.

Stability

- Addressed an issue in the PrizmDoc Server where the Watchdog service was unable to restart the failed microservices after an internal error.
- Addressed an issue in the PrizmDoc Server where the PCCIS module of PrizmDoc Viewer failed with a "System.OutOfMemoryException" error when processing large documents that were referenced as a URL when creating a Viewing Session or uploaded with `PUT /PCCIS/V1/ViewingSession/u{viewingSessionId}/SourceFile` API. Also PrizmDoc Server memory consumption was significantly reduced for scenarios mentioned above.
- Improved the cache clean-up algorithms of PrizmDoc Server. Previously, it became permanently unhealthy

due to cache corruption with the MongoDB.

- When installing or uninstalling the PrizmDoc Server for Windows, the checkbox **Restart Now** on the final page of the product's Installer is now checked by default if the installer made modifications that require system reboot, such as [registry changes](#).
- The Docker image [accusoft/prizmdoc-application-services](#) base has been updated to Ubuntu 18.04.
- Addressed an issue in the PrizmDoc Server where the ms-office-conversion-service might not restore Microsoft Word, Excel, or PowerPoint instances that had crashed.
- Addressed an issue in the PrizmDoc Server where the child LibreOffice instances of the Office Conversion Service (OCS) might continue to consume CPU resources in the background, while the service is idle. This used to happen after viewing or conversion of large documents rendered by the PrizmDoc Viewer as a plain text.
- Addressed an issue in the PrizmDoc Server where load balancer may crash when `GET /admin` request is sent during the product start.
- Addressed an issue in the PrizmDoc Server with enabled MSO rendering engine where it either failed start, or was not fully functional on the machines with more than 32 CPU cores. On such systems, initialization of a large number of MSO instances either took too much time, or did not complete successfully at all - this is related to the system resources usage by MSO. Now the PrizmDoc Server limits the max CPU core usage to 32 for Office documents processing, so if you need to utilize more CPU power for Office documents viewing and conversion, please consider using [PrizmDoc Server Clustering](#).

Security

- Updated bundled version of Java included in our product to a newer version of AdoptOpenJDK (1.8.0_282), taking advantage of recent security fixes and other changes in the JRE.
- Updated MongoDB included in our product to version 4.2.14 on Windows platform, taking advantage of recent security fixes.
- Updated the client-side viewer and samples to the latest official version of Underscore (v1.13.1), taking advantage of the most-recent updates and bug fixes.
- Updated the client-side viewer and samples to the latest official version of jQuery (v3.6.0), taking advantage of recent security fixes.

Release Notes v13.16

PrizmDoc Viewer v13.16 introduces the following:

- [New Features & Improvements](#)
- [Product Updates & Fixes](#)
- [Documentation Updates](#)

13.16 New Features and Improvements

- **Metered Licensing customers can now view their usage.** [Accusoft Customer Portal](#) now provides the ability for PrizmDoc Viewer Metered Licensing customers to download and view their usage statistics for processed documents.
- **Content Conversion Service (CCS) can now convert XLS and CSV files to XLSX.** If you are using [PrizmDoc Server with Microsoft Office](#), you can now use the Content Conversion Service to convert XLS and CSV files to XLSX. See the [API guide](#) for more information.
- **Experimental option to render meeting information in email files.** PrizmDoc Server has a new experimental [central configuration](#) option for rendering meeting information associated with email files: `__experimental.fileTypes.email.renderMeetingInfo`. See [central configuration](#) for more information. **NOTE:** *This feature is a work-in-progress that is not officially supported by Accusoft. Its behavior may change at any time in a future release of the product. We are collecting and reviewing any feedback you can provide about this feature at <https://ideas.accusoft.com/ideas/PDV-I-745>.*

13.16 Product Updates and Fixes

Viewing

- Addressed an issue in the viewer UI where using the text selection tool (for redactions, highlights, and text selection) sometimes selected unintended characters when starting the selection in the middle of a word.
- Addressed an issue in the viewer UI where clearing a search or executing a new search did not always remove the previously active and selected search result.
- Addressed an issue where some pages failed to display if a large document was opened in `alwaysUseRaster` mode and scrolled through.
- Addressed an issue where text was not available for search or selection when viewing PDF files containing TTF fonts with Kerning sub-table format 2.
- Addressed an issue in the PrizmDoc Viewer API where using `ViewerControl` `convertToHighlight` or `convertToRedaction` method added marks that did not appear correctly and could not be selected, edited, or reloaded when saved. Additionally, `convertToHighlight` and `convertToRedaction` have been deprecated; we recommend you use `addMarkFromSearchResult` instead.
- Addressed an issue with the PAS `GET /v2/viewingSessions/{viewingSessionId}/restrictions` REST API where it would fail with an HTTP 500 error if the viewing session had been created from a viewing package.

Redaction

- Addressed an issue with the PrizmDoc Server `redactionCreators` REST API where it was unable to create redaction definitions for PDF files containing TTF fonts with Kerning sub-table format 2.
- Addressed an issue with the PrizmDoc Server `redactionCreators` REST API for Linux and Docker platforms where, if the source document contained certain 4-byte-long UTF-8 characters, the redactions produced after those symbols were incorrectly shifted.
- Addressed an issue in the [PAS MarkupBurner](#) and [PrizmDoc Server MarkupBurner](#) REST APIs where they were unable to redact specific PDF files. This happened when the PDF document contained either specific inline images, some complex objects, or an empty content stream in the Contents array of the page.
- Addressed an issue in the [PAS MarkupBurner](#) and [PrizmDoc Server MarkupBurner](#) REST APIs where sending a JSON body with a request `Content-Type` of `application/json` with an explicitly defined charset (such as `Content-Type: application/json; charset=utf-8`) would fail.

Text Extraction

- Addressed an issue where text could not be extracted from PDF files containing TTF fonts with Kerning sub-table format 2.

Fidelity

- Addressed an issue when rendering Excel files with MS Office would sometimes render too few or too many pages.
- Addressed an issue where some Ink Annotations in PDF documents were not being rendered.
- Addressed an issue in the [Content Conversion Service \(CCS\)](#) where it produced a text-searchable PDF file with an unexpected page orientation. This happened when the source PDF contained raster images with the `Rotate` property.
- Addressed an issue in the [Content Conversion Service \(CCS\)](#) API causing it to incorrectly apply DPI when converting a document to TIFF with G4 compression, resulting in low-quality TIFF output.
- Addressed an issue in MSG rendering when an email was sent by a software system on behalf of someone else. Previously, PrizmDoc Server would render the "From" field with the name of the software system instead of the name the person who had sent the email for MSG files.
- Addressed an issue where PrizmDoc Server's automatic format detection incorrectly concluded that a text file was a bitmap when the text file began with "BM" or "BA".

- Addressed fidelity issues when rendering specific DICOM images.

Stability

- Improved the MS Office rendering engine to properly detect a broken MS Office installation. Previously, if the MSO rendering mode was enabled, the ms-office-conversion-service declared itself as "running" on the Admin Page even if it was unable to start any MS Word, Excel, or PowerPoint instances.
- Improved PrizmDoc Server's MS Office rendering engine to detect and gracefully fail when converting Office documents that require some sort of human interaction via a popup window in the MS Office application. Previously, processing of such documents was preventing other concurrent Office document conversions from successful completion.
- Addressed an issue with the Office Conversion Service (OCS) intermittently crashing and restarting when using a hybrid of Linux and Windows clusters for MSO rendering and there were processing delays of more than 5 minutes in the Windows cluster.
- Addressed an issue in PrizmDoc Server on Windows where one of the microservices might have mistakenly terminated other microservices during peak load periods, making PrizmDoc Server unhealthy and unable to restart.
- Addressed an issue where viewing document pages with widths greater than approximately 3500 mm or 21000 pixels in `alwaysUseRaster` mode caused the Raster Conversion Service worker threads to hang, occupying up to 10 CPU Cores.
- Addressed an issue where viewing specific GIF images caused PrizmDoc Server to become unhealthy.
- Addressed an issue with running multiple Redaction Creators in concurrent requests.
- Addressed an issue where the PrizmDoc Server "ms-office-conversion-service" log files were not being rotated, resulting in large log files which consumed unnecessary disk space.

Security

- Addressed vulnerability when rendering specific JPEG, TIFF, DICOM and SGI images.

13.16 Documentation Updates

- Updated the [Customizing the Styles](#) topic for accuracy and clarity.
- Updated the Troubleshooting section by adding the new [Memory Consumption Issues](#) topic to help you workaround possible unexpected memory consumption by PrizmDoc Java services.

Release Notes v13.15

PrizmDoc Viewer v13.15 introduces the following:

- [New Features & Improvements](#)
- [Product Updates & Fixes](#)
- [Documentation Updates](#)

13.15 New Features and Improvements

- **New Metered Licensing option.** We now offer a new kind of [Metered License](#) which allows you to use all of the features of PrizmDoc Viewer without any limits as long as your license is current. At runtime, the number of documents you process is automatically reported back to Accusoft. And renewal is easier than ever: simply pay to extend your license and your PrizmDoc Server instances will automatically detect your new license expiration date (no new license key to adopt or servers to redeploy). See [Metered Licensing](#) for more information.
- **Deprecation of Cloud Licensing and Node-Locked Licensing.** While currently still supported, with the introduction of Metered Licensing we are announcing the deprecation of the older Cloud License and Node-

Locked License types. Support for these kinds of licenses will be removed in a future release. See [Licensing](#) for more information.

- **New Angular samples on GitHub.** We've published two new Angular samples to GitHub, [one with a .NET backend](#) and [one with a Java backend](#). These samples are deliberately minimal, designed to give developers a clear, concise example of how to use PrizmDoc Viewer in an Angular context.
- **New markup burning REST API option allows you to only include certain kinds of marks.** When making a request to the [Markup Burner](#) API, you can now request that only annotations, redactions, and/or signatures be included.
- **PrizmDoc Server no longer changes the default Windows printer at runtime.** Previously, when using Microsoft Office for rendering, PrizmDoc Server required the Windows default printer be set to "Microsoft XPS Document Writer" and, if necessary, would forcibly change the default printer to be the "Microsoft XPS Document Writer" at runtime. This is no longer the case. While the "Microsoft XPS Document Writer" printer does still need to be installed and available, it no longer needs to be set as the default printer and PrizmDoc Server will no longer change which printer is set as the default.

13.15 Product Updates and Fixes

- Improved logging for the PrizmDoc PDF Processing Service to eliminate redundant warnings about a structure of PDF documents and to help reduce log file size.
- Addressed an issue with the PrizmDoc Viewer public-request-service failing to communicate with external resources (like image references from emails) over HTTPS and making it impossible to view the associated documents.
- Addressed an issue in the ViewerControl where it displayed the page loading indicator improperly when used in the standalone (chrome-less) mode (without the PrizmDoc Viewer Client.)
- Changed the level of opacity when previewing redactions in the PrizmDoc Viewer Client UI. The opacity level in the client viewer UI is now 20%, which is consistent with the default opacity level used for draft (transparent) redactions. Previously the opacity level in the client viewer UI was 50%.
- Addressed an issue in the PrizmDoc Viewer Plain Text Redactor API where redacting a document failed if the markup JSON contained Text Annotations.
- Addressed low-performance issue on Windows when converting or viewing PDF documents that extensively use embedded fonts.
- Addressed an issue with the PrizmDoc Viewer Office Conversion Service (OCS) intermittently failing to start and becoming 'Unhealthy' on the Admin Page in AWS Fargate or other Linux environments with NFS shares when there are network connection delays.
- Addressed an issue in the PrizmDoc Viewer where opening an EML document attachment failed when Content-Disposition header was not defined for the attachment.
- Addressed an issue with the PrizmDoc Viewer on Windows where it could leave dangling wkhtmltopdf.exe processes when the service was running or after shutdown.
- Addressed an issue in the ViewerControl where clicking to dismiss a menu would add a mark if a text-based mark tool is selected. When using the text selection, highlight, strikethrough, hyperlink, or text selection redaction tool, clicking will no longer select text or add a mark. You must now click and drag to select text or add a mark.
- Addressed an issue in the ViewerControl where using the "begins with" or "ends with" search matching option would not find instances of the search term itself. Instances of the search term are now returned when performing a search using the "begins with" or "ends with" search matching option.
- Addressed an issue in the Viewer where multiple comment search results were selected in the search results panel when filtering output.
- Addressed an issue with the PrizmDoc Viewer Workfile Service where it sporadically failed to clean up work files and then bloated logs with error messages on being unable to delete those work files.
- Addressed an issue in the ViewerControl where marks were added in the wrong location on a page. This happened when the `addMarkFromSearchResult` method was used (to add marks from search results), the page was out-of-view, and the page was a different size than the first page.
- Addressed an issue in the ViewerControl where using the `scrollTo` or `scrollToAsync` method did not scroll to the correct position.

- Addressed a vulnerability when rendering specific CSV documents in the LibreOffice rendering mode.
- Addressed an issue with the restart of the PrizmDoc Redaction Service that was caused by an out of memory exception when the `plainTextRedactors` functionality was used for source files containing hundreds of pages.
- Addressed an issue in the PrizmDoc Viewer where its micro service restart was delayed on Windows when `WMIC` was not responding.
- Improved the PrizmDoc Service health detection to allow it to become healthy again as soon as a backend service is back up and running. Previously, the PrizmDoc service would become permanently unhealthy when a backend service could not be restarted within 2 minutes.
- Introduced new parameters in Central Configuration to control Java Virtual Machine (JVM) settings when starting PrizmDoc Server Java-based services (PDF Processing Service and Email Processing Service). For more information, refer to the JVM Options section of the [Central Configuration](#) topic.
- Addressed an issue that allowed "Ends With" and "Begins With" to be selected in the Viewer at the same time.

13.15 Documentation Updates

- Re-organized the [Known Issues](#) section of the Release Notes to make it easier to locate relevant information.
- Updated the Troubleshooting section to make it easier to locate information on [resolving PrizmDoc Server health issues](#) and [log file issues](#). Added content to the [Troubleshooting > Document Viewing Issues](#) section for a workaround when viewing a document with more than 10,000 pages.
- Corrected documentation issues and updated content in supported file formats, working with viewing packages, and code examples.

Release Notes v13.14

PrizmDoc Viewer v13.14 introduces the following:

- [New Features & Improvements](#)
- [Product Updates & Fixes](#)
- [Documentation Updates](#)

13.14 New Features and Improvements

- **PrizmDoc Viewer Markup Burner API and Viewer Client API now provide the ability to apply draft redactions without actually obscuring the content.** The [Markup Burner](#) API and the [Viewer Client](#) API now allow you to produce PDF documents with transparent (draft mode) redactions that display the document content underneath the redaction rectangles.
- **The [Content Conversion Service \(CCS\)](#) now provides the ability to convert documents to raster TIFF using specific DPI (Dots Per Inch) image resolution.** When choosing TIFF as the output file format, the Content Conversion Service (CCS) API now allows you to set a specific resolution on the output raster image - it will have the requested DPI value and will be scaled if necessary.
- **PrizmDoc Redaction Creators API now provides support of multiple redaction reasons.** When creating redactions using the [Redaction Creators API](#), you can now specify [multiple redaction reasons](#) to be associated with redaction marks.
- **New React sample on GitHub.** We've published a new [React sample](#) on GitHub. This sample is deliberately minimal, designed to give developers a clear, concise example of how to use PrizmDoc Viewer in a React context.
- **The ability to retain configuration settings when upgrading PrizmDoc Server.** Starting with version 13.14, when upgrading version 13.3 or higher, PrizmDoc Server is preserving the [server side](#) configuration available in `prizm-services-config.yml` and `pcc.config` configuration files.
- **The ability to retain configuration settings when upgrading PrizmDoc Application Services.** Starting

with version 13.14, when upgrading version 13.8 or higher, PrizmDoc Application Services are preserving the [configuration](#) available in the `pcc.nix.yml` and `pcc.win.yml` configuration files.

- **PrizmDoc Viewer Client API now provides support for opening email attachments in the same viewer window.** You can now configure [PrizmDoc Viewer Client](#) to open email attachments in the same Viewer window where you are viewing the original email.
- **New PrizmDoc Viewer Client UI for browsing email attachments.** PrizmDoc Viewer Client UI has been improved to show email attachments in a compact dropdown menu instead of a panel. Click the paperclip icon to view the list of email attachments and switch between the email and its attachments without shrinking the document viewing area.
- **Improved PrizmDoc Viewer performance when viewing specific CAD-like PDF documents.** PrizmDoc Server has been updated to produce more optimal SVG content for CAD-like PDF documents which contain a lot of small consecutive elements having the same style and transform. This optimization leads to much higher browser responsiveness while zooming, panning, and annotation drawing operations in PrizmDoc Viewer. It also slightly improves the thickness level of the rendered lines making it more accurate on display.

13.14 Product Updates and Fixes

- Addressed an issue in the PrizmDoc Content Conversion Service (CCS) where footer text was misplaced outside of the visible content in the output document. This happened when it was applied to a rotated PDF page that had the Art, Bleed, or Trim boxes defined partially outside of a Media box.
- Addressed an issue in the Markup Burner API where garbage text showed in the output PDF instead of Unicode text. This happened when text annotations were applied with different encodings (ASCII and Unicode) from one markup file for the same embedded font.
- Addressed an issue in the Markup Burner API where redacting a document failed when the source PDF document had a newline character as a delimiter for internal PDF stream elements.
- Addressed a PrizmDoc limitation of the MSO rendering engine causing inability to process Word 95 binary documents and templates.
- Addressed an issue in the PrizmDoc Content Conversion Service (CCS) where the diagonal watermark in the output PDF document was not correctly centered compared to the rendered content in the Viewer.
- Addressed an issue in the Markup Burner API where redacting a PDF document containing specific JBIG2-encoded images would apply over the raster image as expected; however, when the output was saved to PDF, the redacted part of the image was not removed.
- Improved PrizmDoc Viewer Client immediate menu customization. You can now configure the maximum height of the menu as described in the [How to adjust immediate menu size](#) section for a better user experience when working with [multiple redaction reasons](#).
- Addressed an issue in the [Redaction Creators API](#) where the redactions it created were usually too small to show redaction reasons after the burning. The size of rectangle redactions created by the Redaction Creators API is now consistent with the size of the text selection redaction rectangles created in PrizmDoc Viewer.
- The Microsoft Office Conversion connectivity for PrizmDoc Servers running on Linux now performs a retry of the failed operation when the connection to the Windows server/cluster fails. This type of situation may happen upon the recycling of Windows servers in a cloud environment.
- Addressed an issue in the Microsoft Office Conversion connectivity for PrizmDoc Servers running on Linux where the conversion of large Office documents failed if it took more than 2 minutes.
- Addressed an issue in PrizmDoc Server where the PCCIS module of PrizmDoc Viewer consumed a lot of memory and would be terminated by OOM on the Linux platform. This happened when processing large documents (about 380 MB and more) that were referenced as a URL when creating a Viewing Session.
- Addressed an issue with rendering specific email (EML) files in the Viewer that contained HTML body content with a character set that was not specified in the Content Type field which prevented users from seeing the content of such email files.
- Addressed an issue with rendering nested MSG attachments from specific MSG files in the Viewer which prevented users from seeing the attachments. Now when you view the emails, you can also view the attachments.
- Support for Internet Explorer was deprecated with PrizmDoc Viewer v13.14. Support for Internet Explorer

will be removed in PrizmDoc Viewer v14.0 which is currently planned for mid-late 2021.

- Addressed an issue in the PrizmDoc Viewer Client UI where the Redact Full Pages dialog was displaying an incorrect list of default multiple redaction reasons. The incorrect list was based on a previous list of redaction reasons that were used for the Rectangle Redaction or Text Selection Redaction.
- Addressed an issue in the PrizmDoc Viewer services (when running on Linux and using a PrizmDoc Cloud License), that caused the services to fail after a system reboot.
- Addressed an issue in ViewerControl that caused an error during burning request generation and didn't allow users to burn in documents with more than 2000 pages from PrizmDoc Viewer Client UI.
- Addressed PrizmDoc Viewer WorkFile service crash when uploading about one thousand files concurrently. This update has improved the throughput of the concurrent uploads of small files (up to 50 KB) up to 3 times.
- Addressed an issue in the Markup Burner API where redacting a document containing internal PDF stream with text and new line characters would apply over the document as expected; however, when the output was saved to PDF, the text was not removed.
- Support for CentOS 6 is deprecated with the release of PrizmDoc Viewer v13.14, as [CentOS End of Lifetime](#) was announced for November 30th, 2020.
- Addressed an issue in the Markup Burner API where redacting a document with a text selection redaction over text smaller than 8 points would display the redaction reason text in the viewer as expected; however, when the output was saved to PDF, the redaction reason text was removed.
- Addressed an issue in the PrizmDoc Viewer Client where search result highlight was not restored after Text Selection Redaction, Highlight Annotation or Strikethrough Annotation was created from the toolbar.
- Addressed an issue in the PrizmDoc Viewer, where Redaction Creators API did not redact all occurrences of a text string in the document, when the extracted text of the document contains NULL (U+0000) characters.
- Addressed an issue in the PrizmDoc Viewer Client where a search result message was incorrect, when the search was using wildcards and had incorrect term.

13.14 Documentation Updates

- We improved the [Troubleshooting](#) section by reorganizing the content and adding a list of technical questions and solutions to help you identify and troubleshoot the most frequently occurring issues.

Release Notes v13.13

PrizmDoc Viewer v13.13 introduces the following:

- [New Features & Improvements](#)
- [Product Updates & Fixes](#)
- [Documentation Updates](#)

13.13 New Features and Improvements

- **PrizmDoc Viewer Client API now provides support of multiple redaction reasons.** When creating a redaction in the PrizmDoc Viewer Client UI, you can now apply [multiple redaction reasons](#) to be associated with the selected redaction. These reasons will be visible in the Viewer and saved to PDF along with the rest of the redaction content.
- **The [Content Conversion Service \(CCS\)](#) now provides the ability to convert documents to 1-bit raster, 8-bit grayscale, 8-bit indexed or 24-bit RGB raster TIFF.** When choosing TIFF as the output file format, the Content Conversion Service (CCS) API now allows you to set a Bitonal, a Grayscale, an Indexed or a RGB color mode to convert every page of a document to a 1-bit raster, a 8-bit grayscale, an 8-bit indexed or a 24-bit RGB raster image.
- **The [Content Conversion Service \(CCS\)](#) now provides the ability to convert documents to raster TIFF using LZW, G4 and JPEG compressions.** When choosing TIFF as the output file format, the Content Conversion Service (CCS) API now allows you to set a LZW, a G4 or a JPEG compression type to convert

every page of a document to a compressed raster image.

- **Improved PrizmDoc Viewer performance when viewing multi-page DWF documents.** In this release we significantly reduced the conversion time and memory consumption when viewing multi-page DWF documents containing 10 or more pages. The performance gain to display such documents is roughly proportional to the number of pages and can be 10 times and more comparing to the previous version of the product.
- **Improved PrizmDoc Viewer performance when retrieving revision data for Document Compare feature** which produces hundreds or thousands of differences between original and revised documents.
- **Improved PrizmDoc Viewer client responsiveness when viewing and scrolling through document comparison results** containing hundreds or thousands of differences between original and revised documents.

13.13 Product Updates and Fixes

- Addressed an issue in the PAS create-tables tool which, when exporting an initialization SQL script for MySQL, produced a .sql file which was not directly executable due to a missing semicolon.
- Addressed an issue for Linux users using the MSO rendering engine. PrizmDoc Viewer was not able to display a document when it was reloaded and viewed again after the workfile lifetime interval expired (the workfile lifetime interval is specified by the [workFiles.lifetime](#) central configuration parameter).
- Addressed a vulnerability when rendering specific Excel documents in the LibreOffice rendering mode.
- PrizmDoc Application Services (PAS) Amazon S3 storage provider now supports loading credentials by assuming an Identity and Access Management (IAM) role via an OpenID Connect (OIDC) web identity token file.
- Addressed an issue where environment variable expansion on Windows was not being applied for the `logs.path` property in the PrizmDoc Application Services configuration file.
- Addressed an issue with the LibreOffice-based rendering engine where MS Word documents with paragraphs that had the line spacing rule set to "Auto" and also had contextual spacing enabled, were rendered with incorrect spacing between the lines.
- Addressed a problem with opening specific MSG files in the Viewer that contained HTML embedded into the RTF body (with RTF encoding, code pages: cp20127 and cp50220).
- Addressed an issue in the ViewerControl where the CSS style that is defined for the HTML body element caused the Rectangle Redaction, Text Annotation, or Text Signature to be displayed incorrectly.
- Updated the Client-side Viewer to hide the "Select" and "Cancel" immediate menu items by default when creating an annotation or redaction. To add these menu items back, use the [immediateActionMenuActionsFilter](#) property.
- Addressed incorrect behavior within the PrizmDoc Application Services (PAS) Viewing Session API. Previously, when creating a Viewing Session for a source URL which responds with a non HTTP-200 status code, it was rendering HTTP response body content indicating a successful creation of the viewing session, instead of reporting an error. With this update, the PAS will now return an error, ensuring that a Viewing Session will not be successfully created due to incorrect source URL.
- Addressed an issue where the environment variable expansion on Windows was not being applied correctly to paths in the PrizmDoc Application Services (PAS) config file when a path contained more than one environment variable.
- Addressed vulnerability issues when rendering specific ICO, PNG, TIFF, DICOM and XBM images.
- Addressed a race condition in the PCCIS module of PrizmDoc Viewer. Previously, when the original and revised documents were uploaded for the comparison process in parallel, it caused the Viewing Session to return a `CouldNotCompareDocuments` error.
- Addressed an issue where a sequential upload of original and revised source files to the PrizmDoc Application Services (PAS) comparison viewing session was taking longer than expected due to a mutual lock.
- Updated [PrizmDoc Cloud Licensing](#) to properly handle the case when the prizmdoc-server container uses a subset of the host's logical cores, specified via the CPU affinity mask. This allows the use of Cloud Licensing for running multiple prizmdoc-server containers on the same host.
- Addressed an issue with incorrect, less than expected "Content-Length" header value returned by the

PrizmDoc Application Services (PAS) Viewing Session restrictions API.

- Updated the PrizmDoc Viewer startup logic on the Windows platform to check whether the system's non-interactive heap size corresponds to the CPU core count and report the product as 'Unhealthy' on the Admin Page in case of a discrepancy. This will help you know immediately when there is a server configuration error. Please visit the [Registry Changes](#) page for more information.
- Updated the PrizmDoc Viewer Client UI to automatically adjust the redaction reasons dropdown height, and thus minimize the scrolling through the list of reasons.

13.13 Documentation Updates

- Improved ability to view methods in the [Viewer API](#) by displaying them in the right-hand navigation panel.

Release Notes v13.12

PrizmDoc Viewer v13.12 introduces the following:

- [New Features & Improvements](#)
- [Product Updates & Fixes](#)
- [Documentation Updates](#)

13.12 New Features and Improvements

- **Improved performance of PrizmDoc Server** on servers which have 8 or more cores by adjusting the number of concurrent PDF conversion processes according to the number of available cores.
- **Improved stability of PrizmDoc Server** when concurrent requests for viewing complex PDF documents exceed the server capacity by making the PDF Conversion Service skip conversions for timed out requests and survive the excess load, instead of becoming unhealthy and restarting.
- **Content Conversion Service (CCS) now provides the ability to apply diagonal text watermarks when converting to PDF.** When choosing PDF as the output file format, the [Content Conversion Service](#) (CCS) API now allows you to apply a diagonal text watermark to every page of the output document.
- **New Evaluation Docker Image.** The new [prizmdoc-viewer-eval](#) Docker image provides a simple PrizmDoc Viewer backend and demo, making it easy to evaluate the product on a single machine.

13.12 Product Updates and Fixes

- Addressed an issue in PrizmDoc Viewer that caused an unlicensed version of PrizmDoc Viewer Self-Hosted to suppress notifications about the evaluation limitations in IE browsers.
- Addressed an issue in ViewerControl that caused an error during markup deserialization when `creationDateTime` or `modificationDateTime` field didn't have milliseconds (a second fraction) specified.
- Addressed an issue in PrizmDoc Viewer where an annotation layer selected for merge was not showing a checkmark indicating it was selected.
- Addressed an issue in PrizmDoc Viewer which caused an error when merging annotation layers if some of the layers referenced pages that were not yet loaded by the viewer.
- Addressed an issue in the Markup Burner API where redacting a page with the use of the `TextSelectionRedaction` mark would apply the redaction mark over the area and remove text as expected but did not remove raster, vector, or hyperlinks content from the burned PDF output of the redacted document.
- Addressed an issue in the MSO rendering engine with inconsistent rendering of non-trustworthy digital signatures as trustworthy.
- Addressed an issue in PrizmDoc Viewer services that caused PrizmDoc Viewer to duplicate or skip pages when rendering TIFF images with old-style JPEG encoding and inconsistent StripOffsets and

JPEGInterchangeFormat tags.

- Addressed an issue in PrizmDoc Viewer services that caused PrizmDoc Viewer to render 2-bit LZW-compressed TIFF images incorrectly on Windows.
- Addressed a vulnerability when rendering specific PNG images.
- Legacy PHP Sample has been removed from PrizmDoc Viewer.
- Addressed an issue in PrizmDoc Viewer services where PrizmDoc Viewer could not display untitled attachments in EML files.
- Addressed a DOM-based XSS vulnerability when viewing documents in PrizmDoc Viewer.
- Addressed an issue in PrizmDoc Viewer that caused a viewing session watermark text to be rendered 30% larger than the expected font size.
- Addressed an issue in ViewerControl where the character encoding information of the annotation and redaction markup being saved was not specified as UTF-8 in the Content-Type HTTP header for the HTTP request causing the text to be incorrectly interpreted.
- Addressed a potential Java workflow vulnerability when viewing or processing specific documents in PrizmDoc Viewer.

13.12 Documentation Updates

- The online help has been updated with a new look & feel for easier navigation which includes: a left panel which outlines the section you are viewing, a right-side "mini table of contents" in each topic to help you jump to the information you need instead of scrolling, and a web-inspired search bar at the top.
- The [Getting Started](#) section has been updated so it's easier for you to evaluate PrizmDoc.
- We've added the [Initial Integration](#) section to help you easily transition to the next step after evaluation.
- The [Administrator Guide](#) section has been reorganized so that you can find the information you need quickly and easily. The section contains everything you need to install, license, configure, cluster, and troubleshoot the [PrizmDoc Server](#) and [PAS](#) backend services.
- Added a table to the [PAS Configuration](#) topic that provides a list of storage entities and supported storage providers.
- Improved introductory content for the [PAS](#) and [PrizmDoc Server](#) REST APIs.
- Updated [Security Guidance](#) with recommendations on avoiding the Server Side Request Forgery vulnerability.

Release Notes v13.11

PrizmDoc Viewer v13.11 introduces the following:

- [New Features & Improvements](#)
- [Product Updates & Fixes](#)
- [Documentation Updates](#)

13.11 New Features and Improvements

- **Docker Images** - In addition to our traditional installers, we are now making PAS and PrizmDoc Server available as Docker images, making setting up a PrizmDoc Viewer backend dramatically easier. Refer to the following topics for more information:
 - [Evaluation with Docker](#)
 - [Deploying PrizmDoc Server with Docker](#)
 - [Deploying PAS with Docker](#)
- **PrizmDoc Server .NET SDK** - For .NET developers doing backend document processing with PrizmDoc Server, we now offer an official [PrizmDoc Server .NET SDK](#) as an open source NuGet package.
- **Installer Updates** - The PrizmDoc Server installer now allows in-place upgrades so that you no longer need to uninstall the previous version of PrizmDoc Server before installing the newest version.
- **Evaluate PrizmDoc Viewer Self-Hosted** - You can now evaluate PrizmDoc Viewer Self-Hosted without an

installed license, making it easier to evaluate the product. PrizmDoc Server now automatically runs in evaluation mode with a fixed feature set if started without a license. For more information, refer to the [Product Evaluation](#) section of the documentation.

- **PrizmDoc Viewer Raster Conversion Service** - Improved the PrizmDoc Viewer Raster Conversion Service to significantly reduce memory consumption and reduce the time it takes to generate raster tiles when viewing large image files.

13.11 Product Updates and Fixes

- Addressed an issue in the Email Processing Service (EPS) causing PrizmDoc Viewer to fail when converting or viewing MSG files with multiple levels of MSG attachments.
- Addressed an issue with the `PCCViewer.Viewer.destroy()` method throwing an error when it was used before the `ViewerReady` event was fired.
- Re-enabled in-place upgrades for the PrizmDoc Server so that you no longer need to uninstall the previous version of PrizmDoc Server before installing the newest version.
- Addressed an issue in the Raster Conversion Service (RCS) that caused PrizmDoc Viewer to fail when viewing huge 1-bit image files with more than 500,000,000 pixels.
- Addressed an issue in the Markup Burner API where applying `TextSelectionRedaction` markup with redactions located on several pages resulted in applying redaction rectangle mark(s) to the first page only in the PDF output as opposed to applying them to all affected pages as specified in the JSON markup.
- Support for Windows Server 2008 R2 was deprecated with PrizmDoc Viewer v13.7 and will soon be completely removed. PrizmDoc Viewer v13.11 is the final release to claim deprecated support for Windows Server 2008 R2. Future releases will no longer support Windows Server 2008 R2.
- Support for Ubuntu 14.04 LTS was deprecated with PrizmDoc Viewer v13.6 and will soon be completely removed. PrizmDoc Viewer v13.11 is the final release to claim deprecated support for Ubuntu 14.04 LTS. Future releases will no longer support Ubuntu 14.04 LTS.
- Addressed an issue in PrizmDoc Viewer services that caused PrizmDoc Viewer to render tiled TIFF images incorrectly, showing only the top left tile.
- Addressed an issue in PrizmDoc Viewer services that caused PrizmDoc Viewer to render 2-bit LZW-compressed TIFF images incorrectly on Linux.
- Addressed a vulnerability when rendering specific malformed TIFF, PNG, BMP and GEM images.

13.11 Documentation Updates

- Updates to the [Administrator Guide \(Self-Hosted\)](#) section:
 - [New overview content](#)
 - New topics about deploying with Docker:
 - [PrizmDoc Server with Docker](#)
 - [PAS with Docker](#)
 - [Minimal Backend Quick Start](#)
- Replaced the "Get an Evaluation License" section with the new [Evaluating](#) section which describes how PrizmDoc Viewer Self-Hosted can be used for evaluation.

Release Notes v13.10

PrizmDoc Viewer v13.10 introduces the following:

- [New Features & Improvements](#)
- [Product Updates & Fixes](#)
- [Documentation Updates](#)

13.10 New Features and Improvements

- **New viewing option!** Accusoft has built an advanced spreadsheet viewer called [PrizmDoc Cells](#) to give PrizmDoc Viewer users the ability to review Excel files exactly as they would appear in the native application. You can analyze formulas, view charts and graphs, view multiple spreadsheets in a single workbook, search for content, and navigate without pagination. For more detailed information on how to integrate PrizmDoc Cells, click [here](#).
- **Improved PrizmDoc Viewer performance when viewing specific CAD-like PDF documents.** PrizmDoc Server has been updated to produce more optimal SVG content for specific CAD-like PDF documents so that browsers will not freeze while rendering the content. The improvements lead to much higher browser responsiveness while zooming, panning, and annotation drawing operations in PrizmDoc Viewer. This optimization also slightly improves the thickness level of the rendered lines making it more accurate on display.
- **Updated client-side viewer to the latest version of jQuery.** We have updated the client-side viewer and samples to the latest official version of jQuery (v3.4.1), taking advantage of the most-recent updates and bug fixes.

13.10 Product Updates and Fixes

- Improved the download filename when a PAS viewing session displayName does not include a file extension. PrizmDoc Viewer will now add an automatically generated extension to the name specified by displayName if it does not include a file extension.
- Removed potential for broken PAS viewing session when displayName does not include a file extension.
- Addressed an issue in the Text Extraction service causing PrizmDoc Viewer text search to fail when viewing PDF documents containing Type 3 fonts that specify default font metrics and attributes using a font descriptor dictionary.
- Updated PrizmDoc Viewer services to predictively generate fully optimized SVG content ("`svgb`") when viewing files from the Chrome browser, instead of the partially-optimized SVG content ("`svga`"). This improves responsiveness of the viewer in the Chrome browser and reduces disk space usage on the server.
- Removed PrizmDoc Server RPM package dependency on `urw-fonts` containing incorrect configuration which led to inappropriate font substitution for the 'fantasy' font family on CentOS 7.

13.10 Documentation Updates

- The [PrizmDoc Viewer End User Guide](#) is now available in both PDF and Microsoft Word formats on our website so that you can customize and redistribute it to your end users as needed.
- Added the new [Attachments](#) topic to the PrizmDoc PAS API Reference documentation.
- Added the new [How to upgrade PrizmDoc Viewer](#) topic to the Administrator Guide.
- Added the following note to the [Viewing Sessions > POST /PCCIS/V1/ViewingSession > Request Body > file](#) section: > **NOTE:** *By default, "file" is not enabled as a valid documentSource. Enable "file" by adding it to the `viewing.sessionConstraints.documentSource.allowedValues` array in [Central Configuration](#).*
- Updated the [Viewing Sessions](#) topic with a complete list of available URLs in the PrizmDoc PAS API Reference documentation.
- Updated the following topics by adding introductory content: [Administrator Guide](#), [API Reference](#), [PrizmDoc Server Configuration](#), [PAS Configuration](#), and [Security Guidance](#).
- Updated the [ViewerControl.deserializeMarks\(values\)](#) method description with an Example section.

Release Notes v13.9

PrizmDoc Viewer v13.9 introduces the following:

- [New Features & Improvements](#)
- [Product Updates & Fixes](#)

- [Documentation Updates](#)

13.9 New Features and Improvements

- **The Viewer now gracefully falls back to non-optimized SVG when web fonts cannot be used.**
PrizmDoc Viewer makes extensive use of dynamically-generated web fonts to optimize the SVG we send to the browser for viewing. In some customer environments, the browser is forbidden from using web fonts. In these environments, our viewer used to render a document with what looked like “garbled text.” In this update, our viewer now automatically detects if font loading is possible and, if not, gracefully falls back to non-optimized SVG which does not require any fonts.
- **Improved text selection boundaries of OCR conversion output.** Previously, when using PrizmDoc Server’s [Content Conversion Service REST API](#) to OCR a document and produce a text-searchable PDF, the bounding boxes of detected characters in the output document were sometimes too short. If text-based redactions were later applied to the output PDF document, the selected text may not have been entirely removed. With this update, the Content Conversion Service REST API will now OCR a document and produce bounding boxes of the correct height for detected characters, ensuring that subsequent text-based redactions are properly applied.
- **Updated bundled version of Java, now using AdoptOpenJDK instead of Oracle Java.** We have updated the bundled version of Java included in our product from Oracle Java 1.8.0_181 to AdoptOpenJDK 1.8.0_212, taking advantage of the most-recent security fixes in the JRE and moving to a Java runtime whose license will continue to permit unrestricted redistribution. Starting with version 13.9, Oracle Java will no longer be bundled with the product.

13.9 Product Updates and Fixes

- Addressed an issue in the Markup Burner API where a required font from the PrizmDoc Viewer installation (that is used by the text markup being burned) could not be found. This font initialization issue was specific to a case where the text markup (with that font) was burning for the first time after the PrizmDoc services initialization and resulted in the Markup Burner API failure.
- Addressed a PrizmDoc limitation of the MSO rendering engine (when viewing or converting specific Excel documents with a defined PrintArea), which produced an incorrect output with the default value of `fileTypes.excel.renderOnlyPrintArea` central config parameter.
- Addressed an issue within the Markup Burner API that was causing an unexpected grey background in the output PDF after burning in a Rectangle Redaction. This could occur when a Filled Rectangle redaction was applied over a PDF page that had a JPEG 2000 compressed image and the transparency area was set by the image mask (i.e a mask entry in the image dictionary).
- Addressed an issue in PrizmDoc Viewer which caused documents to display unreadable text when web fonts were disabled in the browser or ad blocker settings. PrizmDoc Viewer will now switch to a fallback mode in this case, which allows displaying of the document text correctly. Please note that fallback mode for disabled web fonts will result in slower document rendering and scrolling, so we highly recommend to keep the web fonts enabled in the browser.
- Addressed an issue in the PrizmDoc Content Conversion Service (CCS) (when performing optical character recognition (OCR) to convert a raster file to a searchable PDF document) that resulted in the height of the recognized text within the PDF output to be smaller and not match its image counterpart.
- Addressed an issue in the PrizmDoc Content Conversion Service (CCS) (when performing optical character recognition (OCR) to convert a GIF file with an unspecified resolution to a searchable PDF document) that caused the resulting PDF document to have incorrect page rotation.
- The "hosting options" dialog has been removed from the PrizmDoc Viewer client installer. The PrizmDoc Viewer client installer now always uses a Self-Hosted hosting option. Please see [Choosing a Backend Hosting Option](#) for more information about different options for hosting the backend.

13.9 Documentation Updates

- Updated the topic to clarify [How to Enable Content Encryption in the Viewer](#).
- Updated the [PAS Configuration](#) topic with additional code examples under the Configuring Storage section.
- Updated the topic [Work with Viewing Packages](#) to clarify the Raster Content and Watermark properties.
- Updated all of the Linux and Windows installation and Licensing topics to include the following: > **NOTE:** *If you have an updated license, you must restart PAS and PrizmDoc Server in order to use the new license.*
- Updated the [Adjust Caching Parameters](#) and [Implement Caching Strategies](#) topics with new steps on how to manually delete the cache and the following: > **NOTE:** *If you set the cache to 1 day, the timer will start over if someone accesses a file that is in the cache.*
- Updated the [Getting Started](#) topic with the following: > **IMPORTANT:** *Scripts must be loaded in the specified order as shown below.*

Release Notes v13.8

PrizmDoc Viewer v13.8 introduces the following enhancements/improvements and product updates/fixes:

- [New Features](#)
- [Product Improvements](#)
- [Product Updates & Fixes](#)
- [Beta Features](#)
- [Documentation Updates](#)

13.8 New Features

- **Support for Windows Server 2019.**
- **New Plain Text Redactors API.** The PrizmDoc Server REST API has long supported the ability to burn-in redactions to a document, producing a redacted PDF via the [markupBurners REST API](#). In this release, we're introducing a [plainTextRedactors REST API](#) which allows you to similarly produce *plain text* output. In the output plain text, a special `<Text Redacted>` placeholder is used to denote that one or more characters of the original plain text has been redacted. Any document you've been redacting with the [markupBurners API](#) can be similarly redacted to plain text with the new [plainTextRedactors API](#).

13.8 Product Improvements

- **New and improved getting started guide.** We've completely re-written our getting started guide, making it easier than ever to understand how PrizmDoc Viewer works and how to integrate it into your web application. Additionally, this new getting started guide explains how you can leverage [PrizmDoc Cloud](#) to accelerate your evaluation, avoiding the need to install any of the backend server-side software when all you want to do is evaluate the viewer.
- **New samples on GitHub.** We've published great new introductory samples for [node.js](#), [ASP.NET](#), and [Java / Spring](#) on GitHub. These samples are deliberately minimal, designed to give developers a clear, concise example of how PrizmDoc Viewer actually integrates with a web application.
- **Client-side viewer resources now available on GitHub.** The client-side viewer resources are now available outside of the "Client Installer." If you need to download the pre-built viewer assets (JavaScript, CSS, fonts, etc.), or if you need to make deep customizations to the viewer UI and rebuild it yourself, you can now get all of these resources on GitHub at <https://github.com/Accusoft/prizmdoc-viewer>.
- **Support for Japanese Reiwa era in OpenDocument file formats as well as in LibreOffice rendering mode.**

13.8 Product Updates and Fixes

- Addressed an issue where the expected cursor did not display when hovering over marks on pages scrolled into view.
- Addressed a low-performance issue with reading PDF documents having hexadecimal characters in their dictionary objects.
- Addressed an issue in the Markup Burner API where redacting a rotated raster image would apply the redaction mark over the image as expected but did not remove the image from the burned PDF output of the redacted document.
- Addressed an issue in the Markup Burner API with the following criteria: if you were redacting a raster image on a PDF page that had been rotated and cropped for display, the redaction would apply over the raster image as expected; however, when the output was saved to PDF, the redacted part of the image was not removed.
- Addressed an issue where JSON markup generated by [PrizmDoc Server Redaction Creator API](#) could not be loaded into the viewer.

13.8 Beta Features

- Added support for controlling the default minimum time for the created viewing package content to remain available via a new property `defaults.viewingPackageLifetime` in [PAS Configuration](#). This is a beta feature that is not officially supported by Accusoft and its behavior can be changed at any time in a future version of the product.
- Added Microsoft Azure Blob Storage support for documents, image stamps, markups, form definitions and viewing packages in [PAS Configuration](#). This is a beta feature that is not officially supported by Accusoft and its behavior can be changed at any time in a future version of the product.

13.8 Documentation Updates

- **PrizmDoc Cloud non-default system configuration values are now documented.** The PrizmDoc Server [Central Configuration](#) documentation and the [PAS Configuration](#) documentation now note whenever Accusoft's PrizmDoc Cloud is using a value which differs from the out-of-box product default and what the custom value is.
- **Removed legacy configuration information from the documentation.** All legacy configuration topics that were deprecated in v13.7 have been removed in this release: Format Detection Configuration & Use, Adjust Vector Conversion Settings for Optimal Performance, and Customize Text File Encoding for PrizmDoc Server.

Release Notes v13.7

PrizmDoc Viewer v13.7 introduces the following enhancements/improvements and product updates/fixes:

- [New Features](#)
- [Product Improvements](#)
- [Product Updates & Fixes](#)
- [Beta Features](#)
- [Documentation Updates](#)

13.7 New Features

- **Support for [Ubuntu 18.04 LTS](#).**
- **JSON support in the [PrizmDoc Server Redaction Creator API](#).** Previously, the Redaction Creator API only created the PrizmDoc legacy XML format for text matching a Regular Expression. This was inconsistent since JSON markup layer is used in all other areas of PrizmDoc. For more information on how to use JSON redaction markup created by the new functionality, review the following topics: [Use the Markup JSON](#)

[Schema and Markup JSON Specification.](#)

13.7 Product Improvements

- **Improved stability of PrizmDoc Server running on Linux** by updating Mono run-time components to address possible unresponsiveness of PCC Imaging Services when loading configuration settings on start up.
- **Improved MSO conversion service's resistance to faulty conversion transactions** by properly detecting and recycling its child processes without having to become unhealthy and restart the service.
- **Improved the consistency of the burn-in operation performed over the text redaction markup for every character regardless of its glyph outline.** Now, a character becomes redacted if the redaction area overlaps the character's glyph area (calculated by width * height) by 40% or more.
- **Improved the cursor behavior in the viewer to better indicate the effect of using the mouse.** In supported browsers, the grab cursor is now displayed when the hand pan tool is selected. The cursor no longer changes when hovering over a mark if the selected mouse tool does not support mark selection. If the selected mouse tool does support mark selection, the move cursor is displayed over a mark only if the mark is selectable.

13.7 Product Updates and Fixes

- Support for Windows Server 2008 R2 was deprecated with PrizmDoc Viewer v13.7 and will soon be completely removed. Future releases will no longer support Windows Server 2008 R2.
- Addressed an issue where the installed samples are not able to build when following the provided README steps due to a missing folder and misplaced assets.
- Addressed the Markup Burner issue with text annotations containing tabulation characters that were incorrectly converted to white square characters in the output PDF upon burn in. We are now replacing tabulations with spaces for more accurate fidelity in the output PDF.
- Addressed a PrizmDoc limitation of the MSO rendering engine causing the conversion process to fail on any PowerPoint document once the MS PowerPoint instance becomes unresponsive.
- Addressed issues where PrizmDoc's file downloading APIs failed to download files which had non-ASCII characters in file names, or corrupted filenames during the downloading. These APIs now use an RFC 8187 compliant Content-Disposition header in their responses to correctly support such file names. The only exception is GET /PCCIS/V1/WorkFile/{fileId}, which continues to use the simpler ASCII-only syntax when using an automatically generated file name. This was done to preserve backward compatibility for customers who use PrizmDoc Client package v13.6 or older.
- Addressed incorrect recycling of MSO conversion service's child processes upon the service's startup, which could result in a faulty conversion transaction making the entire MSO conversion service permanently unhealthy.
- Addressed incorrect location of text redaction markups over bulleted text in PDF output to match what appears in the viewer.
- Addressed an issue within the Markup Burner API that resulted in a duplicate Redaction Rectangle mark in the output PDF. This would occur when applying a Filled Rectangle redaction over a PDF with the transparency area defined by the image mask (a mask entry in the image dictionary).
- Addressed PAS host header translation issue which made it impossible to route requests from PAS to PrizmDoc when there is a load balancer in between them. Now, PAS sets the outgoing request host header to PrizmDoc host value.
- Corrected an issue in the Viewer UI that allowed the user to select the "Exact Phrase" option which is not a viable option for Wildcard search. The Exact Phrase option is now disabled in Wildcard search.
- Updated the Search Tasks API to accept a JSON body of up to 1MB in size. Previously, this value was set to 100KB which could have resulted in "Invalid JSON" errors for valid JSON bodies that were greater than this value.
- Addressed the Markup Burner API issue occurring when attempting to redact text in the text block immediately following a previously redacted block in the source PDF document.

- Addressed an issue where in some cases the last page of a document was blank when printed.

13.7 Beta Feature for Evaluation

- **Content Conversion Service (CCS) now provides the ability to convert PDF documents to MS Word (DOCX) documents.** This functionality requires the [Microsoft Office rendering mode](#) to be enabled by the MSO feature in your license key. This is a beta feature that is not officially supported by Accusoft and its behavior can be changed at any time in a future version of the product, but we are happy to collect and review any feedback you can provide about this feature. The API to convert to MS Word (DOCX) documents should be used for evaluation purposes only and should not be used in production deployments. Note that this API may undergo change prior to feature completion. Please refer to the [API guide](#) for the details on how to evaluate this feature.

13.7 Documentation Updates

- **Legacy PrizmDoc Server configuration documentation has been removed.** This legacy documentation information had been marked as deprecated for many releases. We strongly recommend that you use [Central Configuration](#). If you need to access legacy configuration documentation, please refer to an archived version of the [documentation](#).

Release Notes v13.6

PrizmDoc Viewer v13.6 introduces the following enhancements/improvements and product updates/fixes:

- [New Features](#)
- [Product Improvements](#)
- [Product Updates & Fixes](#)

13.6 New Features

- **JSON support in the [PrizmDoc PAS MarkupBurner API](#) and [PrizmDoc Server MarkupBurner API](#).** Previously, the MarkupBurner API only accepted the PrizmDoc legacy XML format for annotations and redactions. This was not very convenient and sometimes confusing since we use JSON markup layer in all other areas of PrizmDoc. For more information on the new functionality, review the following topics: [Use the Markup JSON Schema](#) and [Markup JSON Specification](#).

13.6 Product Improvements

- **Smoother page scrolling, faster page loading in the viewer.** When a user is scrolling through a document, sometimes pages will delay loading causing the user to wait for the next/previous page to load. With this update, the document scrolling is smooth and next/previous pages load immediately providing a more responsive user experience.
- **Faster conversion with Microsoft Office.** We've rewritten the task scheduler inside the Microsoft Office Conversion Service (MSOCS) to improve performance. The Office Conversion Service uses MSOCS to process its requests when the MSO option has been licensed, using multiple workers that execute in parallel. There are now a larger number of workers in both services, keeping more documents resident in memory simultaneously, while continuing to have the same amount of execution concurrency (so as to not overwhelm the server). This means that document reloading by the workers happens less frequently, gaining performance overall.
- **Improved PDF fidelity.** Improved PDF rendering service to address multiple PDF rendering fidelity issues including, but not limited to: more accurate color rendering, raster images rendering, text rendering

(including math symbols), rendering of linearized PDF documents, as well as rendering of write protected PDF documents.

13.6 Product Updates and Fixes

- Addressed an issue where the Document Compare API detected an inaccurate number of differences in documents containing tracked changes that were unaccepted.
- Addressed a PrizmDoc limitation of the MSO rendering engine causing the conversion process to fail when converting an empty Excel document.
- Modified PrizmDoc logging so that the header and footer data is not recorded in the log files when appending a specified header or footer to a document's pages with the use of PrizmDoc CCS API for enhanced security and user privacy.
- Modified PrizmDoc PAS logging so that email attachment names are not recorded in the log files (when rendering email documents with attachments) for enhanced security and user privacy.
- Addressed PrizmDoc PAS issue that caused the viewingPackageCreator process to hang when uploading the output page artifacts to S3 storage (when the size of the artifacts exceeds 20 MB).
- Support for Ubuntu 14.04 LTS has been deprecated with PrizmDoc Viewer v13.6 and will soon be completely removed. Future releases will no longer support Ubuntu 14.04 LTS.
- Addressed an [XSS](#) vulnerability when displaying the context menu for Text Hyperlink Annotations in the Viewer. Prior to the fix, a malicious script could be injected into the URL value for a Text Hyperlink Annotation and executed when the context menu for the annotation was displayed. The fix ensures that the URL value is properly escaped during construction of the HTML to display the context menu for the Text Hyperlink Annotation in the Viewer.

Release Notes v13.5

PrizmDoc Viewer v13.5 introduces the following enhancements/improvements and product updates/fixes:

- [New Product Name](#)
- [Enhancements & Improvements](#)
- [Product Updates & Fixes](#)
- [Documentation Updates](#)

New Product Name

- **Product name changed from *PrizmDoc* to *PrizmDoc Viewer*.** To accommodate our growing number of products, we have changed the name of this product to *PrizmDoc Viewer*. And we are excited to announce that *PrizmDoc* is now a suite of products which includes *PrizmDoc Viewer* and the all-new [PrizmDoc Editor!](#)

13.5 Enhancements and Improvements

- **Improved visual rendering of 1-bit raster documents** by adding upscale image dimensions preventing quality degradation for files with asymmetric resolutions.
- **Smaller PrizmDoc Server log files.** We have updated PrizmDoc Server logging to eliminate redundant information from console logs, decreasing their size.
- **Support for Unicode filenames in email attachments.** We have improved support for detecting and listing email attachments with names using RFC 8187 encoding in EML documents to provide broader support of email attachment types.
- **Added rendering support for UTF-7-encoded email.** Added rendering for UTF-7-encoded email document content that includes email message body, headers, and attachment names to support a broader range of UTF encoding types.
- **PAS now accepts a full database connection string,** allowing Microsoft SQL Server customers the ability

to enable database-specific features like connection encryption.

- Security improvements:
 - Addressed XXE vulnerability in PrizmDoc Server.
 - Upgraded PrizmDoc Server to the latest version of Java 8 to take advantage of security fixes within the JRE.

13.5 Product Updates and Fixes

- Addressed the issue with all conversions from raster to searchable PDF that failed at the same time (when reaching a certain number of concurrent conversions).
- Addressed incorrect rendering of Pantone colors on Windows when viewing PDF documents.
- Updated PrizmDoc Viewer installation instructions to ensure accuracy (see details below under Documentation Updates section).
- Addressed an issue where PrizmDoc Viewer was unable to load OpenXML Word documents (containing invalid external file references) when running in LibreOffice rendering mode.
- Resolved an issue where burning a document using XML annotations for hyperlinks and setting the fillColor value to 0, for black, would instead render the hyperlink in a default blue color.
- Addressed PrizmDoc Viewer email processing service, PDF processing service and MS Office converter logging to rotate the corresponding log files according to the logging.daysToKeep setting in the Central Config file.
- Addressed an issue to prevent freehand annotations with a width and height of 0 from being created. Previously, a user could create an empty annotation that would cause errors when the document is downloaded and opened in Adobe Viewer.
- Updated ViewerControl API double-click handling to use the dblclick event (instead of the mouseup event and a timer) to fix issues with double-click handling in IE v11.

13.5 Documentation Updates

The "What's New?" section is now called Release Notes

- The "What's New" section has been renamed and updated to include sections for Product Updates/Fixes, Enhancements, and [Known Issues](#) (formerly located in the Release Notes on the website).

New Topics

- [What's New? > Known Issues](#)

Updated Topics

- [Troubleshooting > PrizmDic Server Health Issues](#)
- [PrizmDoc Overview > Legal > Third-Party Attributions](#)
- Fixed misspelling in topic: [API Reference > PrizmDoc E-Signature Viewer API > Module: event-store](#).
- Removed extra characters that were not needed in topic: [API Reference > PrizmDoc E-Signature Viewer API > Module: template-name-header](#).
- Fixed incorrect formatting in topic: [API Reference > Viewer API > PCCViewer.Mark](#).

Release Notes v13.4

PrizmDoc Viewer v13.4 introduces the following enhancements/improvements and product updates/fixes:

- [Enhancements & Improvements](#)
- [Product Updates & Fixes](#)

13.4 Enhancements and Improvements

- **Improved stability under load.** Conversion performance has been enhanced to avoid system degradation in high-volume periods.
- **Lower CPU usage when converting large Microsoft Excel documents.**
- **Faster conversion of Microsoft Office documents.** We've optimized the system to more efficiently convert multiple Office documents, even in periods of heavy usage.

13.4 Product Updates and Fixes

- Addressed an issue where vector content was not completely removed when a document was redacted and downloaded as a PDF.
- Addressed a PDF redaction issue that was causing the grid lines (representing tables in the PDF document) to disappear.
- Addressed potential XSS vulnerability concerns with the Viewer.
- Addressed potential XSS vulnerability concerns with the PDF Processing Service.
- Addressed a PDF redaction issue that was causing parts of a document to disappear.
- Fixed an issue where images within a document were having redactions applied to them when they should not been applied. This occurred when burning redactions into documents on machines using LibreOffice and when the redaction overlaid any portion of a referenced image (on a page within the document) in which the image was not displayed.
- Addressed an Office document conversion issue (running in the LibreOffice rendering mode) to disallow non HTTP and HTTPS protocols when rendering content of WEBSERVICE formula in Excel and OpenDocument Spreadsheet documents.
- Addressed a rendering fidelity issue with semi-transparent PDF elements that were previously rendered opaque.
- Addressed a rendering fidelity issue with PDF highlighter annotations that were previously rendered opaque.
- Updated Office Conversion Service to fix a bug related to document affinity that would sometimes prevent a worker from switching to a new document at the correct time. This would result in new tasks waiting until all work on a particular document was completed.
- Addressed a problem with internal links that were not clickable in a recurring Word document footer when rendered using LibreOffice mode.
- Addressed a PDF redaction issue that caused parts of a document to disappear.
- Significantly improved loading time for certain PDF documents created by a third-party recognition server previously resulting in a page load timeout.
- Addressed a rendering fidelity issue with MSG files (with an RTF body) causing an extra line of text to show up in the message body.
- Addressed an issue with missing libjpeg dependency in PrizmDoc Viewer RPM package when installing on the CentOS 7 platform.
- Resolved an issue in handling miter operators used within PDFs that could cause some redactions to fail to burn into the document.
- Addressed an issue with the rendering of email files that contain Rich Text body (with embedded content) stored in raw binary format.

Release Notes v13.3

PrizmDoc Viewer v13.3 introduces the following new features and product updates/fixes:

- [New Features](#)
- [Product Updates & Fixes](#)

13.3 New Features

Improved Rendering of Microsoft Excel Documents

- **New Excel rendering and pagination options.** PrizmDoc Viewer can now paginate and render Microsoft Excel documents while preserving the original page dimensions and margins specified in the document (matching the Page Layout rendering mode of Microsoft Excel). This is a non-default rendering option, which needs to be explicitly turned on. For more information, refer to the [Central Configuration](#) topic.

Improved Customization and Integration

- **Easier initialization of the viewer.** We've updated the Viewer's gulp build to produce a new `viewerCustomizations.js` that contains the required customization objects to instantiate the viewer control. This allows you to build the customizations once and then easily integrate them into your web app, regardless of the server-side language you're using. Previously, our Viewer samples dynamically built these objects at runtime in the particular server-side language of the sample. This approach made it hard to integrate this code into your own web application (especially if you used a language not covered by our samples).

Improved Office Conversion Service Task Scheduler Performance for LibreOffice Users

- **Faster conversion of Office documents.** We've rewritten the task scheduler inside the Office Conversion Service (OCS) to improve performance. OCS processes its requests with multiple workers that execute in parallel. There are now a larger number of workers, keeping more documents resident in memory simultaneously, while continuing to have the same amount of execution concurrency (so as to not overwhelm the server). This means that document reloading by the workers happens less frequently, gaining performance overall.

13.3 Product Updates and Fixes

- **Improved redaction area text wrapping.** Previously, text that did not fit within the redaction area overflowed beneath the redaction upon downloading (burning). Now, when downloading a document, PrizmDoc Viewer will wrap the text, breaking the line on the space characters in the text and attempting to center the text both vertically and horizontally within the redaction area. When there is a single line of text without spaces, it will truncate the characters of the line of text that do not fit. All of these alterations are designed to mirror the behavior of the redaction reason text seen in the Viewer when viewing, previewing, or printing.
- Added instructions to the online help for how to customize Excel document view settings to match rendering in PrizmDoc Cloud.
- Implemented rendering for inline attachments embedded in the binary HTML body of MSG email files.
- Addressed incorrect rendering of email body content and attachments stored as an encapsulated message with the syntax of "RFC 822 message".
- Updated the MongoDB service, used internally for text searching, to respect the Central Configuration `cache.directory` property. The old cache location can be safely deleted to free disk space if desired:
 - The Windows directory is: `C:\Prizm\services\mongo-manager-service\bin\mongodb\data`
 - The Linux directory is: `/usr/share/prizm/services/mongo-manager-service/bin/mongodb/data`
- Improved the cache cleanup algorithm to be more predictable.
- Modified the MongoDB installation that we use internally to no longer use the default password.
- Addressed incorrect page count calculation specific to the rendering of Excel files (with the disabled "Show Page Breaks" display option) using MSO rendering mode.
- Increased concurrency of Office Conversion Service when processing tasks with LibreOffice on Linux

platforms. Response times now degrade more gradually after the service load reaches full capacity.

Release Notes v13.2

PrizmDoc Viewer v13.2 introduces the following new features and product updates/fixes:

- [New Features](#)
- [Product Updates & Fixes](#)

13.2 New Features

Security Features

- **PrizmDoc Server now supports TLS 1.1 and TLS 1.2 for outgoing HTTPS requests.**
- **Option to disable rendering of externally-referenced HTML content.** We've added a server-side configuration option called, [security.htmlRendering.blockExternalContent](#), to control whether or not externally-referenced HTML content, such as images and iframes, will be blocked. This option affects any source document file type which uses HTML, including HTML, EML, and MSG.

Full Redaction of Vector Images

- **Support for full redaction of vector images** by removing the vector content completely from the output document.

Online Help

- The [PrizmDoc Cloud License & AWS](#) topic has been updated to better explain this licensing option.

13.2 Product Updates and Fixes

Fidelity

- **Improved rendering of email.** Corrected fidelity issues when rendering plain text EML files with non UTF-8 encoded message body.
- **Improved rendering of Microsoft Word documents using merge fields.** Corrected a fidelity issue when rendering Microsoft Word documents with merge fields used for dynamic document creation. Merge field placeholders would be rendered instead of the actual field data.
- **Improved rendering of Microsoft Word Table of Contents.** Addressed a fidelity issue where incorrect rendering of Table of Contents for specific Microsoft Word files with table of contents titles that do not have heading styles applied.
- **Improved rendering of Microsoft Word figure numbers.** Corrected a fidelity issue when rendering Microsoft Word documents with figure numbers containing automatic sequence number fields.
- **Improved rendering of Microsoft Word documents with chart links to external files.** Corrected a rendering issue where Microsoft Word documents that contain charts with links to external files would not render for viewing or convert to JPEG.

Performance

- **Faster merging of PDF documents.** Improved the performance of merging PDF documents (which now take 4-5 seconds instead of taking up to 10 minutes previously).
- **Faster rendering of Microsoft Word documents under load.** Improved the performance of rendering and viewing Microsoft Word documents with fields by 12% over PrizmDoc Viewer v13.1 for high volume runs.
- **Faster restart time.** Improved PDF Conversion Service restart logic for faster recycle and restoration of its

dependency processes which previously could take as long as 5 minutes.

Security

- Corrected a security issue where PrizmDoc Viewer would capture information that could be sensitive when processing emails and record that information in the PrizmDoc Viewer service log files.
- Addressed a service recovery issue when converting an Excel document with a very high number of rows fails to render (millions of rows).
- Corrected a conversion issue where certain Microsoft Word and PowerPoint documents that were converted to PDF using PrizmDoc Viewer would generate a PDF that had issues with text (so that selecting, copying and pasting into another document would produce partial text of the original document).

Release Notes v13.1

PrizmDoc Viewer v13.1 introduces the following new features and product updates/fixes:

- [New Features](#)
- [Product Updates & Fixes](#)

13.1 New Features

Document Compare

Support for Document Comparison feature for Linux platforms. It is important to note that the Microsoft Word document comparison feature uses the Microsoft Office Conversion (MSO) add-on option for PrizmDoc Server running on Windows and therefore requires the PrizmDoc Server running on Linux to be configured to connect to PrizmDoc Server running on Windows.

Content Conversion Service

OCR option to produce text-searchable PDF. [Content Conversion Service](#) (CCS) now provides the ability to perform optical character recognition (OCR) to convert a raster file or a scanned PDF file to a searchable PDF document. The resulting PDF document will contain the original image and the recognized text in a separate invisible layer, with each text character position matching its image counterpart. This will allow you to search, select and copy the text in the resulting PDF document.

The Content Conversion Service's feature that performs OCR (to convert a raster file to a searchable PDF document) does not support CentOS 6 and Red Hat Enterprise Linux 6 platforms.

Rendering Updates

Email contents now rendered with time zone information. We've updated the rendering of email to support the Date and Time header fields of MSG and EML files with the corresponding time zone. When an end user views emails in the Viewer, they will be able to see what time zone the date and time correspond to.

Performance Improvements

- **Faster first page load times in the viewer.** First page of content in the Viewer loads faster now, especially for large Microsoft Office documents.
- **Faster page loading when scrolling.** Page content loads more quickly when scrolling through a document in the Viewer.
- **Faster retrieval of document text.** Eliminated the occasional slowness in getting document text.

Viewer Touch Experience Improvements

- **Improved touch support.** The Viewer touch experience (for phones, tablets, and other touch devices) has been improved with product updates and behavioral improvements.

PrizmDoc Application Services (PAS)

- **S3 support.** PAS now supports Amazon S3 for storage of all of its artifacts, including viewing packages and annotation layers.

Documentation Updates

With PrizmDoc Viewer v13.1, the PrizmDoc Cloud documentation has been added to the PrizmDoc Viewer help file. You can see the new updates here:

- [Cloud Authentication](#)

13.1 Product Updates and Fixes

Viewer

- **Improved memory usage.** Memory usage in the Viewer has been improved in situations where the same web page is used to view multiple documents without recycling the Viewer.

PAS

- **Stability improvements.** The stability of viewing package creation has been improved due to architectural changes.

PrizmDoc Server

- Addressed incorrect rendering of Chinese/Taiwanese date format fields within Excel documents when running through LibreOffice rendering mode.
- Addressed problem with opening specific Outlook Email attachments with explicitly specified "filename" fields as .msg.
- Addressed problem with opening specific MSG files (with raw HTML body) in the Viewer.

Release Notes v13.0

PrizmDoc Viewer v13.0 introduces the following new features and product updates/fixes:

- [New Features](#)
- [Product Updates & Fixes](#)

13.0 New Features

Document Comparison Overview

Comparison rendering of two Microsoft Word documents. We now provide the ability to compare Microsoft Word documents. You can now create a viewing session using two different Microsoft Word documents as input and see a single comparison view of the two documents in the viewer. This comparison view will note whenever text has been added, changed, or removed, when formatting as changed, etc., just as Microsoft Word does with its "Track Changes" feature. For more information on how to use the new Document Compare functionality, refer to the following topics:

- Overview of [Document Compare](#)

- [Work with Document Comparison](#) Programmatically
- [Perform Document Comparison](#)

Image Tools

Image tools in the viewer. We've added new image tools to the viewer, allowing you to adjust things like contrast or line darkness right in the browser.

Support for New Operating Systems

- Support for [Ubuntu 16.04 LTS](#).
- Support for [Windows Server 2016](#).

Native SVG Icons

Native SVG icon support. The Viewer has been upgraded to support native SVG icons. This will simplify replacing default icons with your own versions.

Online Help

- The [Deployment Licensing](#) section has been updated and clarified to help you understand all of your licensing options. A new topic covers the purpose of the Prizm Licensing Utility.

13.0 Product Updates and Fixes

PrizmDoc Server

- Addressed incorrect page count and rendering content issues with MS Word documents (with track changes turned on when using the MSO rendering engine).
- Resolved incorrect rendering of AAA/AAAA Excel date/time formatting when using the LibreOffice rendering engine.

Known Issues

The following items are either currently under investigation by the Accusoft Engineering organization or provide further information regarding PrizmDoc Viewer. Should you require an updated status on any of these items, please contact [Accusoft Customer Support](#).

- [Installation & Upgrade Issues](#)
- [Browser-Specific Issues](#)
- [PDF Issues](#)
- [CAD Issues](#)
- [Office Issues](#)
- [E-Signature Issues](#)
- [Annotation & Redaction Issues](#)
- [Miscellaneous Viewer Issues](#)
- [Content Conversion Service Issues](#)
- [Miscellaneous Server Issues](#)

Installation and Upgrade Issues

- Currently, the path PrizmDoc Viewer is installed to on Windows cannot exceed 64 characters total. Longer paths are rejected by the installer.
- When installing PrizmDoc Viewer on Windows, the account used to start the PrizmDoc service requires a

password to be defined. Without a password, the Installer will not be able to proceed.

- The ext3 file system limits the number of subdirectories within a single directory to 31,998. Because PrizmDoc Viewer creates a directory for each viewing session and/or work file, systems with high traffic combined with longer cache expiration periods may encounter request errors with ERROR_GEN_FAILURE. The ext3 file system is the default for ephemeral drives in AWS as well as many current Linux distributions. Consequently, the possibility of encountering this error exists for these systems unless ext4 was specifically chosen at installation. To enable directories containing greater than the 32K subdirectory limit, ext4 turns on HTree indexes (a specialized version of a B-tree) by default. For systems with extraordinarily high traffic coupled with extended cache expiration times, it is recommended the product be installed on Linux systems with ext4 file systems or at least to configure caching to utilize an ext4 file system.
- If you are using Viewing Packages in PAS and upgrading from a version prior to 13.1, you must upgrade your database schema. See the setting up your database topic for more information on how to upgrade PAS.
- PrizmDoc Viewer relies on a fontconfig package that is not shipped with the product and that might be missing from some distributions of Ubuntu. This was resolved by adding automated checks in the PrizmDoc Viewer 13.0 installation scripts. As a workaround for older versions of PrizmDoc Viewer, we recommend installing the fontconfig package manually by using `sudo apt-get install fontconfig`.
- You must have Microsoft Office installed on the server when using the Microsoft Office Conversion license. If you do not have Microsoft Office installed, you will get an error when converting Office documents or displaying them in the Viewer.
- Always keep your PrizmDoc Viewer Windows server updated with the latest Windows and MS Office updates when using the MSO rendering option. If you don't have the latest updates, you may see rendering issues.

Browser-Specific Issues

- The Safari browser may truncate long file names (containing 96 or more ASCII characters or 27 or more non-ASCII characters) when downloading source files from PrizmDoc viewer. This issue only exists when connecting to PrizmDoc Viewer via http/1.1 protocol (with or without https). Setting up an http/2 proxy in front of the PrizmDoc Viewer avoids the issue. See also the Safari bug report [47914517](#).
- If you print large documents (100+ pages) from the Viewer, you may run into browser memory constraints that are beyond our control. We recommend downloading large documents as a PDF and then printing them.
- When printing a document in Firefox or Safari, embedded images may be truncated or missing in some cases.
- When printing a document in Chrome, images may be printed with a black background when the background should be transparent.
- When printing documents with the Viewer in the Safari browser on Windows, blank pages are sometimes created, causing extra pages in the document.

PDF Issues

- Border of graphic elements may not render for specific PDF documents.
- Starting from 13.6 release some PDF documents may render slightly shifted compared to previous versions of PrizmDoc.
- Raster images using CMYK colorspace in PDF documents may be rendered with a slightly different color.
- Currently, burning markup into PDF documents that contain XFA fields is unsupported. Attempting to burn markup into XFA documents will result in an error.
- Burning markup into PDF documents that contain AcroForm fields is unsupported. Attempting to burn AcroForm fields in PDF documents will leave those field values unchanged.
- PDF files with embedded raster images in the Indexed color space using CMYK palette might not display with the correct colors in the Viewing Client after conversion to SVG.
- Search results returned in the Viewing Client for PDF documents may not be highlighted in cases where the

PDF contains image over text results. In this case, content will be returned in the Search results tab, but the highlighted search terms will not be displayed in the page view when navigating to the appropriate page. In this case, a message will be displayed indicating that the page does not support text highlighting. This will be improved in future versions.

- PrizmDoc Viewer does not currently perform text-extraction and search on PDF annotation markup within PDF documents. This annotation markup includes Text and FreeText annotation types as specified in the PDF reference guide.
- PrizmDoc Viewer does not support rendering of Dynamic XFA (XML Forms Architecture) PDF forms. Instead of the actual forms content, a message such as "Please wait... If this message is not eventually replaced by the proper contents of the document, your PDF viewer may not be able to display this type of document." will be displayed.
- If the PDFPS microservice exceeds the open file descriptor limit, then PrizmDoc Viewer will restart the PDFPS microservice.
- CMYK JPEG files may display as negative (or inverted) color images when converted to PDF or raster formats.
- Deep Image Redaction may not work as expected when the source PDF contains G32D encoded TIFF images. PDF redactions will be created but any content on the G32D encoded content intended for redaction will not be obscured by black pixel data.
- When converting HTML to PDF using the Content Conversion Service, you may see poor performance on larger documents when `pdfOptions.forceOneFilePerPage` is set to true. We recommend that `pdfOptions.forceOneFilePerPage` be set to false when converting more than 20 pages.

CAD Issues

- Embedded OLE objects in CAD files are not currently supported for rendering.
- Watermarking is not supported for CAD files.
- The MarkupBurner, responsible for the underlying process of annotation, redaction and e-signature, does not currently support CAD based files.

Office Issues

- Excel files may take more time to process when using [PrizmDoc Server with Microsoft Office](#) version 2019. Increased processing time depends on the source document content, and may take up to three times longer compared to Excel conversion performed by Microsoft Office version 2016.
- The Document Compare feature is designed for comparing different revisions originating from the same MS Word document. Using this feature for comparing any random MS Word documents is not recommended and may lead to unexpected results.
- Internal hyperlinks (like TOC bookmarks) within Office documents are not clickable when rendered or converted to PDF with the use of the Microsoft Office Conversion option.
- The following TTC font packages on Ubuntu might conflict with Tunga and Latha font substitution implemented in PrizmDoc Viewer Office Converter causing inaccurate rendering. You may need to uninstall those packages for better font substitution fidelity:
 - fonts-gubbi
 - ttf-indic-fonts-core
 - ttf-bengali-fonts
 - ttf-devanagari-fonts
 - ttf-gujarati-fonts
 - ttf-kannada-fonts
 - ttf-malayalam-fonts
 - ttf-oriya-fonts
 - ttf-punjabi-fonts
 - ttf-tamil-fonts
 - ttf-telugu-fonts

- Excel worksheets will now be rendered with grid lines, headers, footers, and hidden content visible by default causing the existing annotation and redaction markup to be not aligned with the old rendering content. Customers wanting to redact the new output would need to re-create annotation and redaction markups. Customers wanting to go back to the old style rendering can do that by changing the Excel rendering properties available in the central configuration file.
- Excel pagination causes the Office converter to generate more pages. This puts more stress on the server and may cause the conversion to timeout.
- Certain VML shapes from Word documents might not render properly to the client viewers.

E-Signature Issues

- With some browsers the E-Signature Viewing Client performance could be impacted when the number of fields in the document is more than 1,300.
- There are known issues with E-Signing when working on iOS devices that can cause the screen to move erratically when moving from field to field. This does not keep the experience from being usable, but it can be disconcerting while using.

Annotation and Redaction Issues

- There are known limitations when trying to use Quick Actions to redact large search result sets that are over hundreds of results.
- Note that we have modified the default properties for several marks when creating them via the API. As long as you are setting mark properties, such as color and line width, this should not affect your code.
- There are limitations to using the Full-Page Redactions mouse tool on a mobile device.
- The Email Conversion Service fidelity improvements will result in shifting the rendered content (due to inline image rendering support for more accurate rendering of the HTML body) and making the existing annotation and redaction markup not aligned with the old rendered content. Customers wanting to redact the new output would need to re-create the annotation and redaction markups.
- There are some special symbols, such as '@', that cannot be properly processed in auto-redaction.
- Burned annotations are shifted on PDF files with a non-standard PDF page Cropbox or Mediabox boxes.

Miscellaneous Viewer Issues

- When requesting raster page content, a '500' error may appear in the network log. As part of performance improvements, the Viewer now always requests SVG content first. If SVG content is not available, a '500' error will be communicated and the client will then request raster content. Although this does not negatively affect the Viewer behavior, this will be changed in a future release to handle the request differently.
- It is recommended that the document cache be cleared prior to upgrading PrizmDoc Viewer. Failure to clear the cache will result in the inability to search documents in the Viewer that have been cached in prior versions of PrizmDoc Viewer.
- PrizmDoc Viewer does not currently provide full support for searching and extracting text which reads from right-to-left (like Arabic and Hebrew), which may cause the search and text extraction results to display such text in wrong left-to-right direction.
- If you print large documents (100+ pages) from the Viewer, you may run into browser memory constraints that are beyond our control. We recommend downloading large documents as a PDF and then printing them.
- Watermarks appear in bold and are not transparent.
- You may see space between the image tiles (when rendering raster images that are broken into tiles, within HTML tables) if those tables do not fit into one page.
- HTML conversion on Windows depends on system DPI. Higher system DPI results in smaller HTML content size when viewing HTML.
- When printing from within the Viewer using a browser other than Google Chrome, you will have to manually

set the page orientation and page size in the system's Print Dialog, even when a particular orientation and/or page size has been set within the Viewer.

Content Conversion Service Issues

- The Content Conversion Service does not support conversion of DICOM files to a searchable PDF document format.
- The Content Conversion Service does not support conversion of transparent TIFF files to a searchable PDF document format.
- The Content Conversion Service's feature that performs optical character recognition (OCR) to convert a raster file to a searchable PDF document does not support CentOS 6 and Red Hat Enterprise Linux 6 platforms.

Miscellaneous Server Issues

- All Server-Side configurations should be identical across all the servers in a cluster because documents are frequently sent to a random server for processing. This also applies for the multi-server Microsoft Office Conversion configuration that now supports connectivity to PrizmDoc Servers running on Linux.
- The POST `/v2/searchContexts/{contextId}/completed` URL should not have a body. Though the request may still succeed if a body is used in the v12.2 release, it can begin to fail in the future. Using any body with this request should be avoided.
- The PrizmDoc Server "GET Page" call requesting a JPEG thumbnail for a certain page hosted on Windows might return HTTP error 500.
- Abandoned Viewing Sessions - The text extraction process will continue if a user abandons a session before the text extraction process completes. This will be improved in a future release, but it is important to know that a user abandoning a document does not necessarily release conversion resources on the server.

Getting Started

This section will help you get started with evaluating PrizmDoc Viewer:

- [Try It!](#) - we'll walk you through the quickest way to try out PrizmDoc Viewer
- [Sample Applications](#) - check out our "Hello Viewer" samples for Node.js, C#, and Java on Github

Try It!

We have a [public Docker image](#) which makes it easy to evaluate PrizmDoc Viewer on a single machine: `accusoft/prizmdoc-viewer-eval`.

Running this image will:

1. Start a demo web application at `http://localhost:8888` where you can easily explore the viewer and its features using your own documents.
2. Start a complete PrizmDoc Viewer backend which you can begin using for local development of your application.

NOTE: The `accusoft/prizmdoc-viewer-eval` Docker image is only suitable for evaluation and local development. It is not suitable for a production deployment. You'll want to use the `accusoft/prizmdoc-application-services` and `accusoft/prizmdoc-server` Docker images instead. See the [Admin Guide](#) for more information.

Minimum Required Hardware Resources

Before running, you **MUST** make sure that you've configured Docker to use the following minimum hardware resources:

- CPUs: **2**
- Memory: **7.50 GB**

If you try to run the image with fewer CPUs or less memory, it may not function correctly.

Starting

First, pull the latest version of the `accusoft/prizmdoc-viewer-eval` image:

```
docker pull accusoft/prizmdoc-viewer-eval:latest
```

Then, start a running container:

```
docker run --rm -p 8888:8888 -p 3000:3000 -p 18681:18681 -e ACCEPT_EULA=YES --name prizmdoc-viewer-eval accusoft/prizmdoc-viewer-eval:latest
```

Optional: Providing a License Key

The example above starts PrizmDoc Viewer without a license key in evaluation mode with a fixed feature set. If you would like to do a full-featured evaluation of the product, please contact info@accusoft.com for a license. If you

have a license, you can provide it by setting two extra environment variables, `LICENSE_SOLUTION_NAME` and `LICENSE_KEY`:

```
docker run --rm -p 8888:8888 -p 3000:3000 -p 18681:18681 -e ACCEPT_EULA=YES -e
LICENSE_SOLUTION_NAME=YOUR_SOLUTION_NAME -e LICENSE_KEY=YOUR_LICENSE_KEY --name
prizmdoc-viewer-eval accusoft/prizmdoc-viewer-eval:latest
```

Just replace `YOUR_SOLUTION_NAME` and `YOUR_LICENSE_KEY` with the actual values.

Wait for the Start-Up Process to Complete

It will take a minute or two for the PrizmDoc Viewer backend to fully start. Eventually, you should see something like this:

```
=====
PrizmDoc Server is running at:
http://localhost:18681

PAS (PrizmDoc Application Services) is running at:
http://localhost:3000

Demo application will be started at:
http://localhost:8888
=====

Starting the demo application...

> prizmdoc-viewer-eval-demo@1.0.0 start /demo
> NODE_ENV=production node main.js

Application running at http://localhost:8888
```

Once you see this, the container is fully up and running.

Using the Demo

After starting, you can view the demo app at `http://localhost:8888` and start exploring the viewer and its features with your own documents.

Stopping

To stop the running Docker container, just use `Ctrl+C` in the same terminal where the container is running.

Alternately, you can stop the container by name: `docker stop prizmdoc-viewer-eval`

Next Steps

Take a look at our [Sample Applications](#).

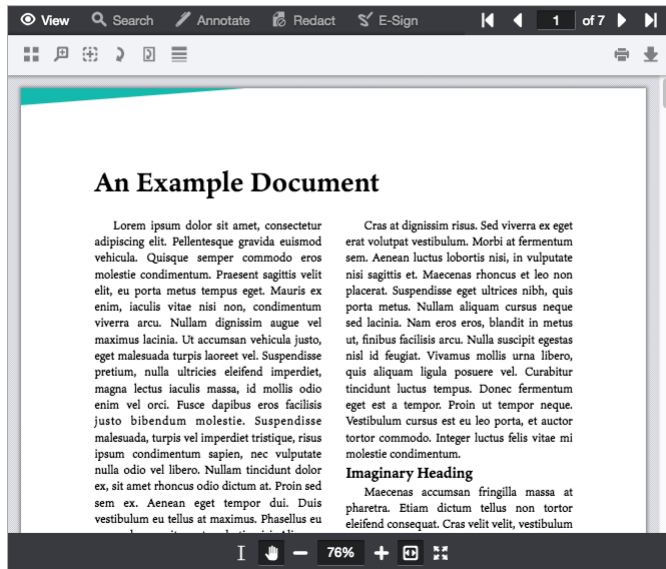
Sample Applications

Our "Hello Viewer" Samples

The "Hello Viewer" samples are introductory samples designed to help you understand the fundamentals of integrating PrizmDoc Viewer into a web application. When they run, they show a simple page with a single viewer displaying an example document, making it easy to see all of the pieces working together:

Hello PrizmDoc Viewer!

This is a minimal node.js express application which loads a document in the browser with PrizmDoc Viewer, like this:



These are extremely simple web applications which do nothing more than load a single document in the browser with PrizmDoc Viewer, but they make it easy to see all of the pieces working together in real code.

Traditional HTML Samples

- [Node.js](#)
- [.NET](#)
- [Java](#)

React Samples

- [Node.js + React](#)

Angular Samples

- [.NET + Angular](#)
- [Java + Angular](#)

Next Steps

For a detailed walk-through of how to integrate the Viewer into an existing web application, move on to [part 1](#) of this three-part guide.

Initial Integration

Introduction

This initial integration guide will walk you through what it takes to integrate a viewer into your web application. When you are through, you should be able to render a viewer in any page of your web application and have it display actual document content.

Let's get started!

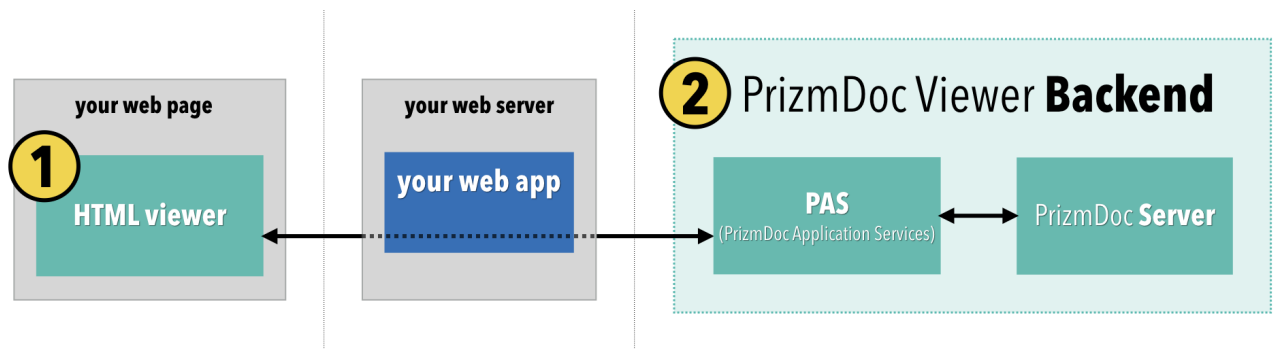
- [Architecture Overview](#)
- [Illustrating the Viewing Sequence](#)
- [1. Integrating the Viewer](#)
- [2. Choosing a Backend and Creating a Viewing Session](#)
- [3. Setting Up the Reverse Proxy](#)

Architecture Overview

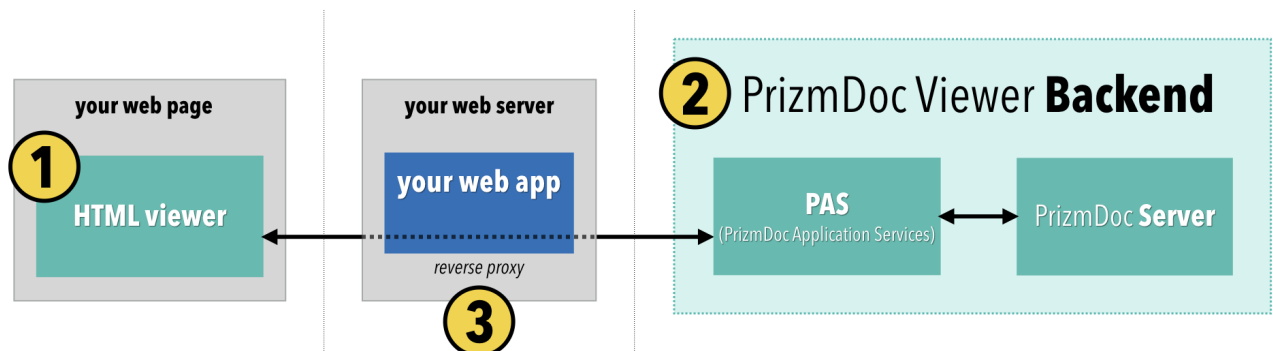
Introduction

There are two main sides of the PrizmDoc Viewer architecture:

1. The HTML **viewer** itself, running in the browser
2. A powerful **backend** which converts document pages to SVG for viewing in the browser



Sitting between these two pieces is your web server, acting as a **reverse proxy** for the viewer to ask the backend for the pages it needs to display:



To integrate PrizmDoc Viewer into your web application, you'll need all three of these pieces:

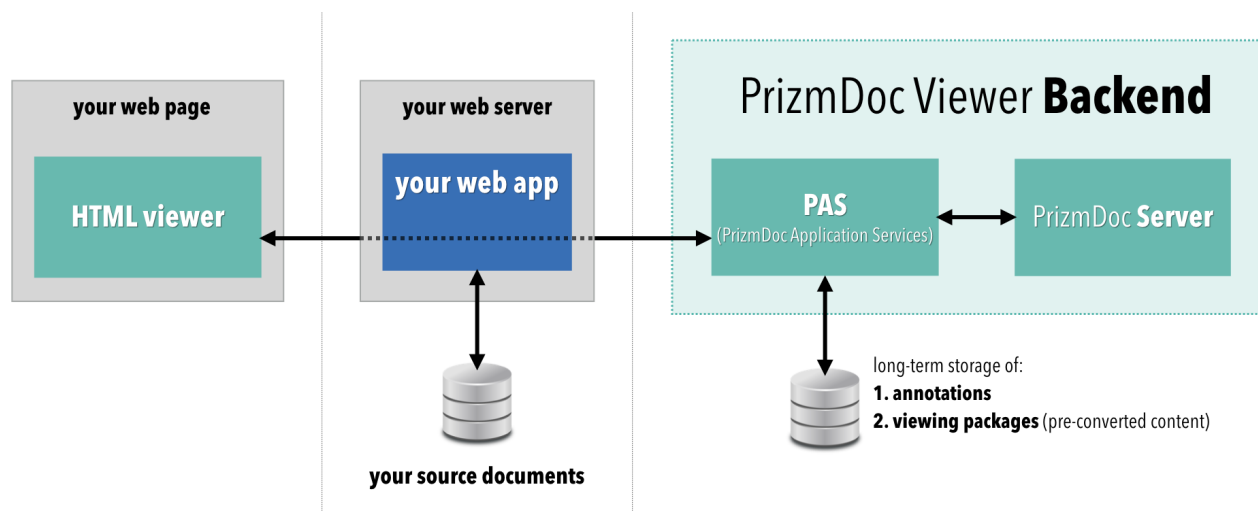
1. the **HTML viewer** itself (a collection of static JavaScript, CSS, font, and image files)

2. a PrizmDoc Viewer **backend**
3. a **reverse proxy** that allows the viewer to request content from the backend

We'll walk you through setting all of this up in the rest of this guide. But first, it helps to have a more detailed understanding of how the pieces work together.

For Viewing, Call PAS

The backend is made up of two tiers, **PAS** (PrizmDoc Application Services) and **PrizmDoc Server**:



PAS and PrizmDoc Server are independent. Each runs on its own host or port, and each has its own REST API.

- **PrizmDoc Server** (on the far right) is the technical heart of the product, the actual engine that converts pages of a document to SVG. It is compute intensive and has no permanent storage.
- **PAS** does not do any conversion work. Instead, it is a layer in front of PrizmDoc Server which is responsible for other viewing concerns, such as saving and loading of annotations or long-term caching of pre-converted content. Like your web application, PAS has privileged access to storage that you own (like a file system or database).

For viewing, your web application should only make REST API calls to PAS. PAS will then make calls to PrizmDoc Server on your behalf to ensure the conversion work is done. The only time your application should call PrizmDoc Server directly is when you need to perform *non-viewing* work, such as converting a file or burning annotations into a document.

Next Steps

Next, let's [take a closer look at how these pieces actually work together when your web application needs to display a document to your user.](#)

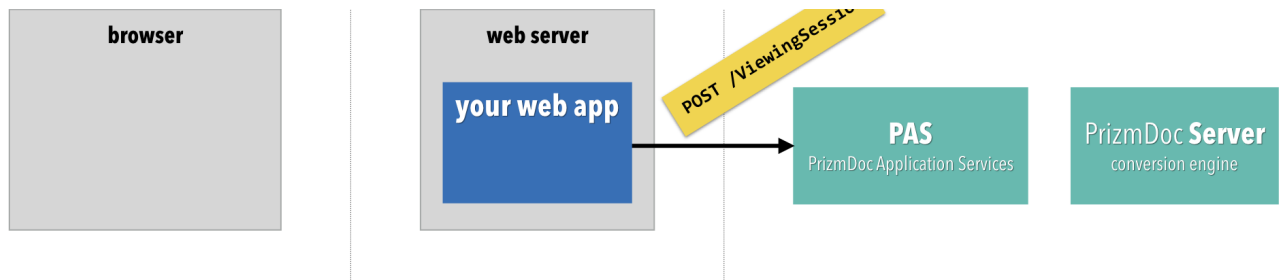
Illustrating the Viewing Sequence

Overview

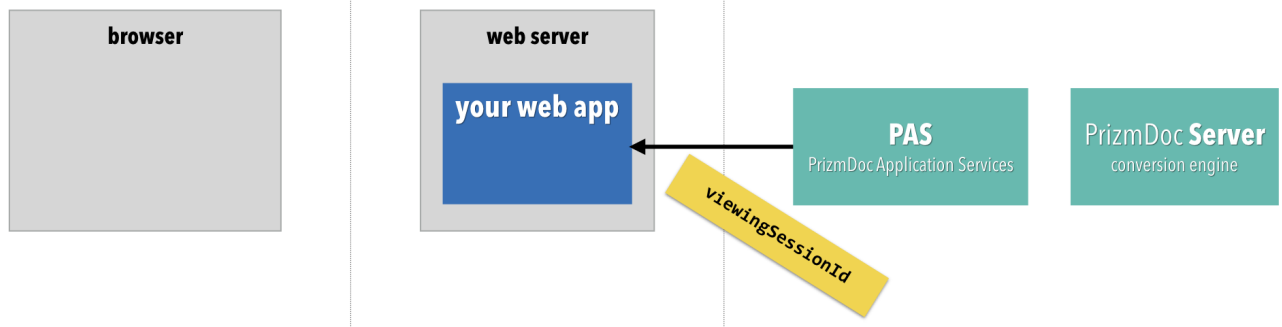
When your web application needs to render an HTML page containing a viewer, there is a sort of “dance” that occurs between the browser and the backend, with your web application sitting in the middle between the two. Here is how it typically goes:

First, your web application POSTs to PAS to create a new *viewing session*:

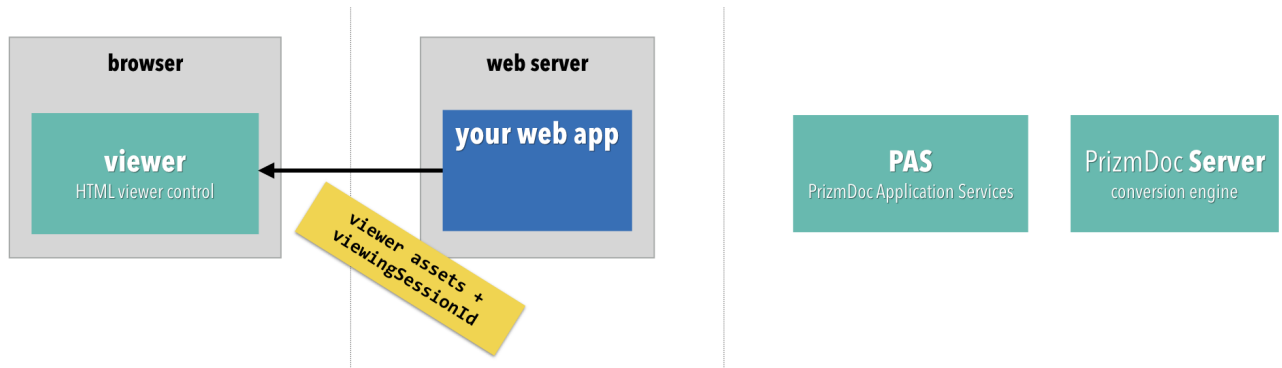




PAS responds to your web application with a new `viewingSessionId`:

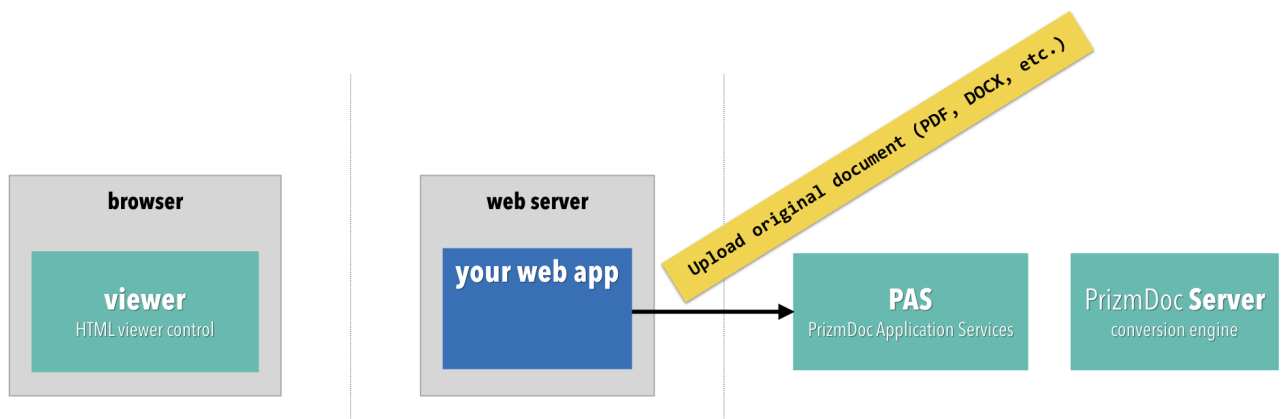


Now, at this point, the document conversion hasn't even started yet. However, you have enough information that your web application can go ahead and render the page HTML with the viewer, configuring it to use the `viewingSessionId` that was returned:



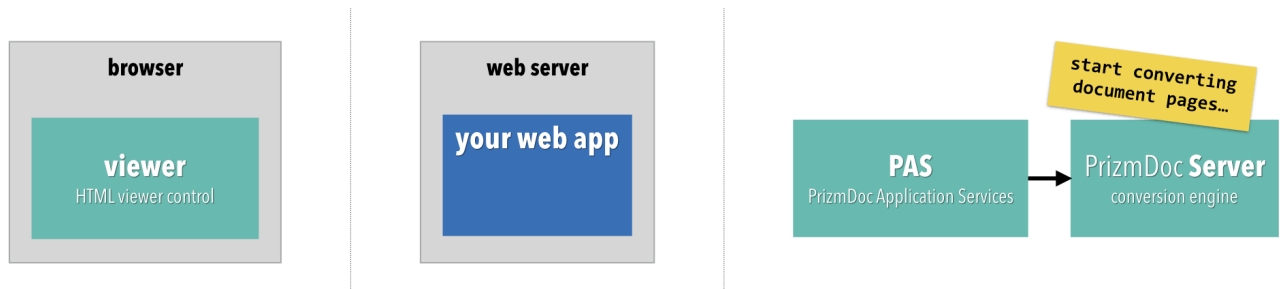
This allows the browser to start rendering the viewer UI as soon as possible.

Next, your web application should upload the original source document to PAS so that the backend can start converting the document:

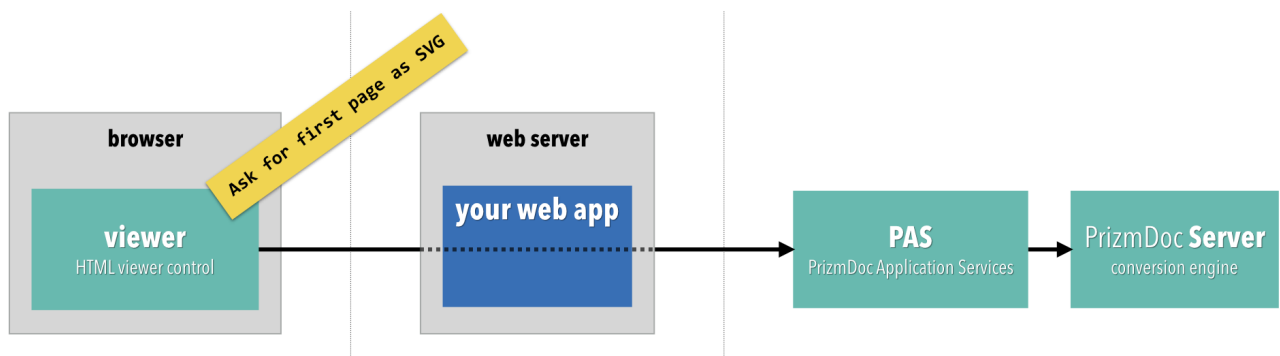


As soon as the source document is received, PAS hands it off to PrizmDoc Server where the document pages are converted, one by one, to SVG. And, as soon as a page is converted, it can be delivered to the browser (even if the

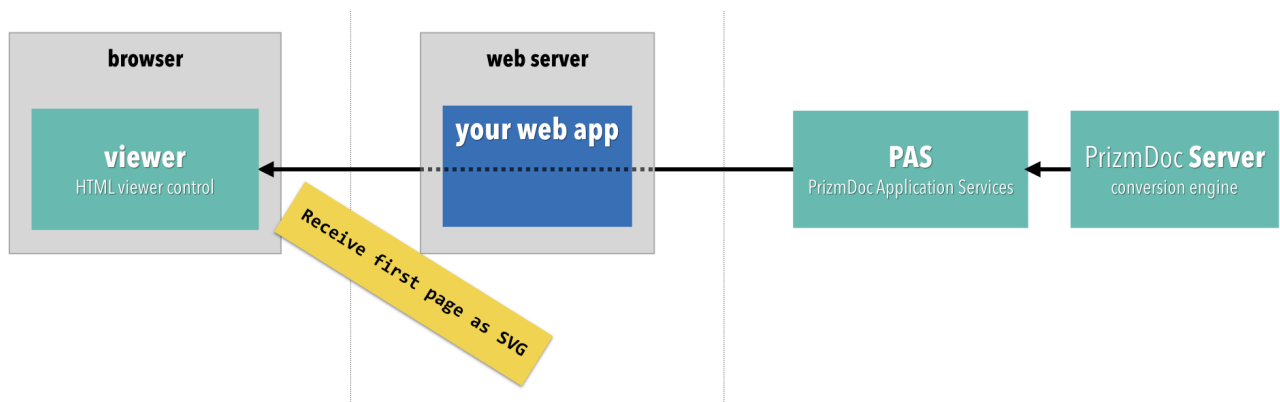
remaining pages are still being converted):



Meanwhile, as soon as it has been loaded in the browser, the viewer begins repeatedly asking PAS (proxied through your web server) for the first page of document content ("Can I have the first page now? Can I have the first page now?"):

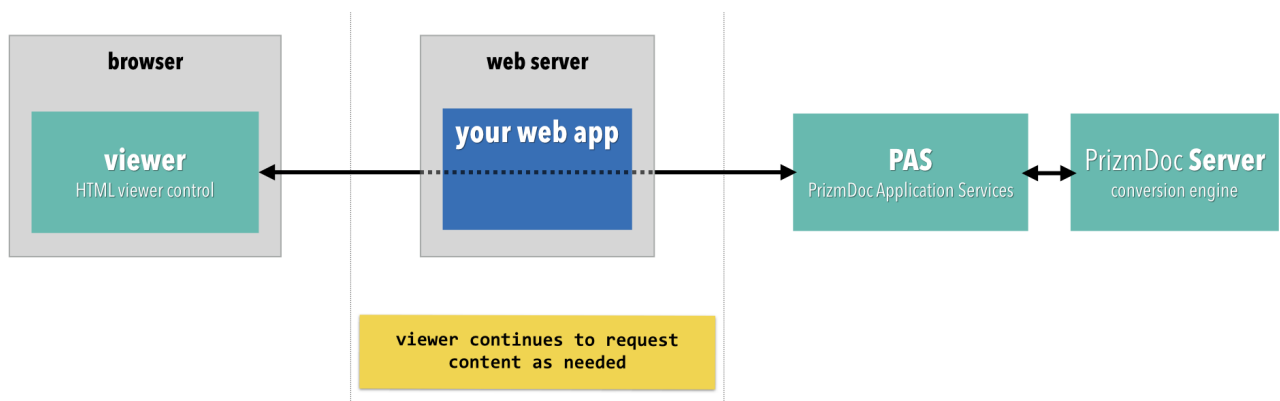


As soon as the first page of SVG content is available, PAS returns it and the viewer displays it to the end user:



Of course, all of this happens very quickly, allowing your users to start viewing and interacting with the document as fast as possible.

As the end user continues to interact with the document, the viewer will make additional requests to PAS for document content as needed:



Next Steps

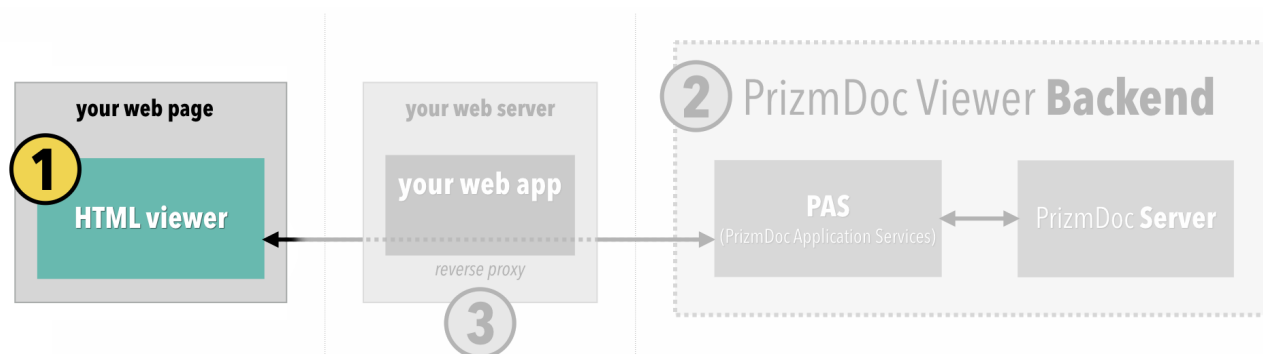
To actually see this happening with real code, [take a look at some of our "hello viewer" sample applications](#). These are great starting points that you can easily set up and run on your own development machine.

If you're ready to start integrating the viewer into an existing web application, move on to [part 1](#) of this three-part guide.

1. Integrating the Viewer

Overview

First, we'll need to add the viewer itself into your web application:



By the end of this section your web application should be able to display an empty viewer.

Update Your Web Application to Serve the Static Files Needed by the Viewer

There are a set of static JavaScript, CSS, font, and image files which define the viewer UI. You'll need to add these to your web application in such a way that the browser can GET them.

You can download the static files needed for the viewer here:

- [viewer-assets.zip](#)

Unzip that file and **move the unzipped viewer-assets directory and all of its contents into your web application at a place where the files can be served statically**. For example, perhaps you configure your web application to map the route `viewer-assets/` to the unzipped viewer-assets directory which you just added to your application. Make sure to include all of the files and folders as defined in the ZIP file, including all of the fonts and images. This is important for all the parts of the viewer UI to display correctly.

Integrate the Viewer Into a Page

To include the viewer in a particular page of your web application, you'll need to:

1. Add boilerplate code in the `<head>` of your page to include the required JavaScript and CSS.
2. Declare a `<div>` somewhere in your HTML as the placeholder for the viewer.
3. Call a JavaScript function which creates the viewer at the location of the specified `<div>`.

1. Include the required JavaScript and CSS

IMPORTANT: Scripts must be loaded in the specified order as shown below.

Assuming you have set up your web application to serve the static JavaScript and CSS files at `viewer-assets/`, you would add the following to the `<head>` of your HTML:

```
<!-- Ensures the viewer works best across various browsers and devices -->
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1
user-scalable=no"/>
<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<!-- CSS required by the viewer -->
<link rel="stylesheet" href="viewer-assets/css/normalize.min.css">
<link rel="stylesheet" href="viewer-assets/css/viewer.css">

<!-- JS required by the viewer -->
<script src="viewer-assets/js/jquery-3.6.0.min.js"></script>
<script src="viewer-assets/js/jquery.hotkeys.min.js"></script>
<script src="viewer-assets/js/underscore.min.js"></script>
<script src="viewer-assets/js/viewercontrol.js"></script>
<script src="viewer-assets/js/viewer.js"></script>
<script src="viewer-assets/js/viewerCustomizations.js"></script>
```

NOTE: If your static files are hosted at a path other than `viewer-assets/`, adjust accordingly.

2. Declare a `<div>` placeholder for the viewer

Somewhere in your HTML, at the location where you want the viewer to appear, add a placeholder div with a unique id, like this:

```
<div id="viewerContainer" />
```

Make sure you style the width and height of the div to be the dimensions you want used for the viewer.

3. Create the viewer

Finally, instantiate the viewer with a JavaScript call like this:

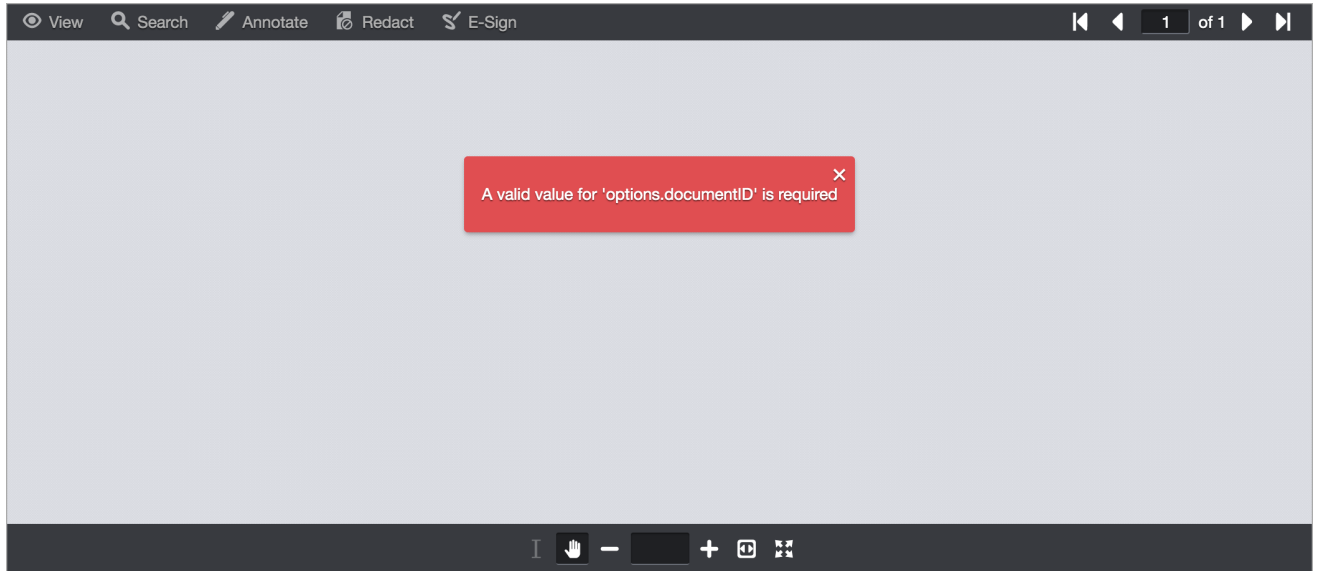
```
<script type="text/javascript">
$(function() {
  $('#viewerContainer').pccViewer({
    //documentID:      'TODO: We'll handle this in part 2',
    imageHandlerUrl:  '/pas-proxy',
    viewerAssetsPath: 'viewer-assets',
    resourcePath:     'viewer-assets/img',
    language:         viewerCustomizations.languages['en-US'],
    template:         viewerCustomizations.template,
    icons:            viewerCustomizations.icons,
    annotationsMode: "LayeredAnnotations"
  });
});
</script>
```

NOTE: If your static files are hosted at a path other than `viewer-assets/`, adjust `viewerAssetsPath` and `resourcePath` accordingly.

You'll notice that the `documentID` has not been set yet. We'll get to that soon.

What We Have So Far

At this point, if you run your web application and view the page in a browser, you should see the viewer UI with the following error:



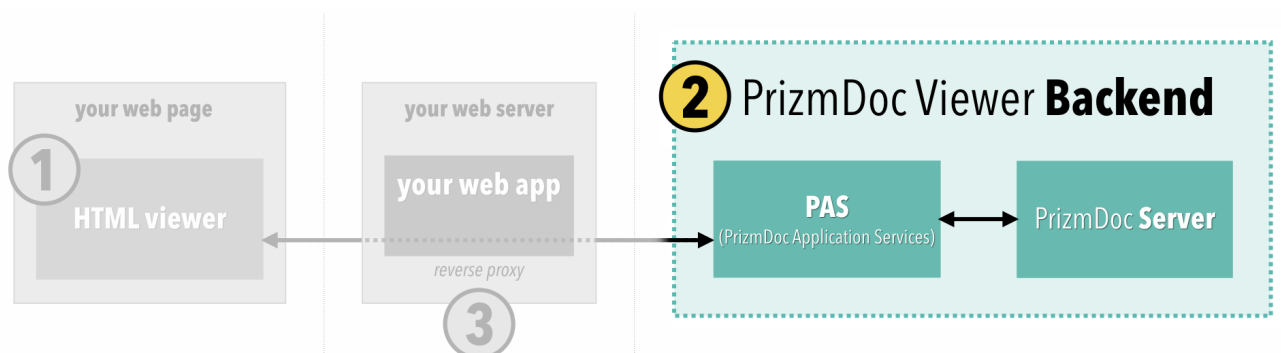
The error is occurring because we have not yet connected the viewer to a backend *viewing session*. We'll do that now in [part 2](#).

2. Choosing a Backend & Creating a Viewing Session

Overview

Next, we need to work with the backend to create a *viewing session*: the "thing" on the backend that is responsible for converting the document and the "thing" the viewer will ask for converted SVG.

In order to do that, we need access to a running backend:



In this section, we will explain:

- the options you have for hosting a backend
- how to create a new *viewing session*
- how to connect a viewer to an associated *viewing session*
- how to provide the source document to the *viewing session*

By the end of this section, the viewer will begin to show a loading indicator (though document content won't load quite

yet).

Choosing a Backend Hosting Option

You have several options for hosting the backend:

1. **Self-Hosted.** You deploy and manage the backend yourself on your own hardware. Data never leaves your network. You pay annually based on the number of machines you run PrizmDoc Server on. For more information, see the [Self-Hosted Backend Administrator Guide](#).
2. **PrizmDoc Cloud.** Accusoft fully-manages a backend which is shared by multiple customers. You pay only for transactions which you use (e.g. documents viewed or processed). *This is the easiest way to get started.*
3. **Private Cloud.** Accusoft deploys a backend for your dedicated use in the cloud and then fully-manages the backend for you. You pay annually for the average number of servers we run for you. This is another easy way to get started. If you're interested in this option, contact info@accusoft.com.

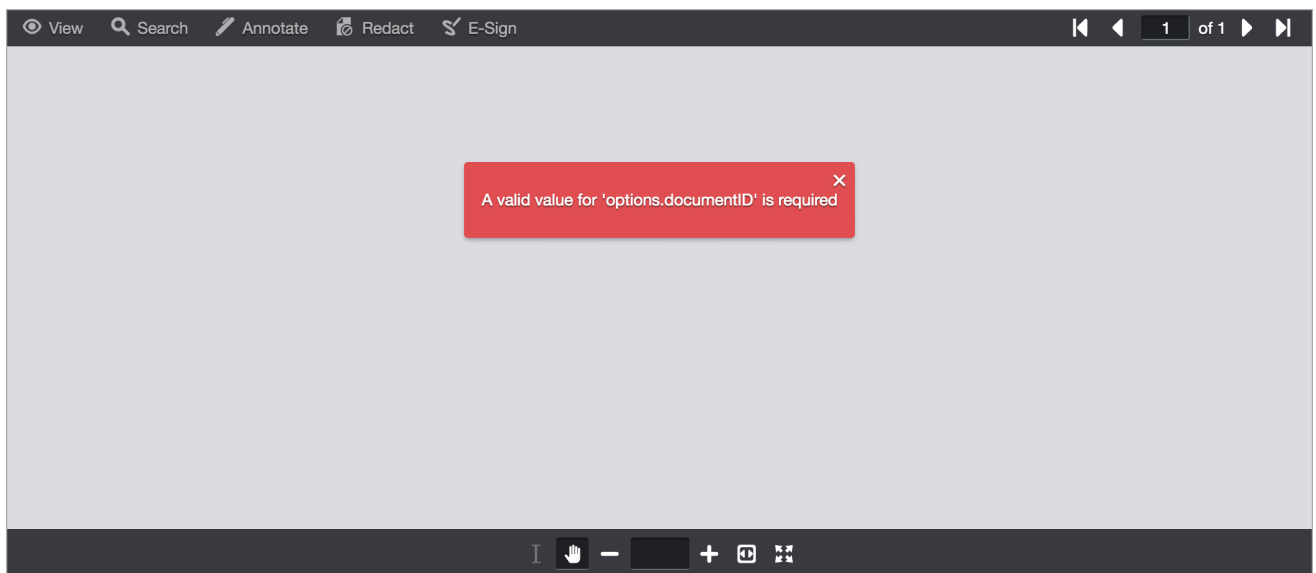
If you're new to setting up a backend, [PrizmDoc Cloud](#) is the easiest way to get started. Just sign up, get your API key, and you're done. Your backend is ready. You can always switch to a private cloud or self-hosted option later if you need to.

About Viewing Sessions

Every viewer instance running in a browser must be associated with a backend *viewing session*. In other words, **any time you create a new viewer, you must first create a new backend *viewing session***.

The *viewing session* is the "thing" on the backend that is responsible for converting the original document to SVG for viewing in the browser, and it is the "thing" the viewer talks to in order to ask for document content ("Hey backend, I need page 9 for *viewing session* XYZ...").

This is why, at the end of [part 1](#), the viewer was giving us an error:

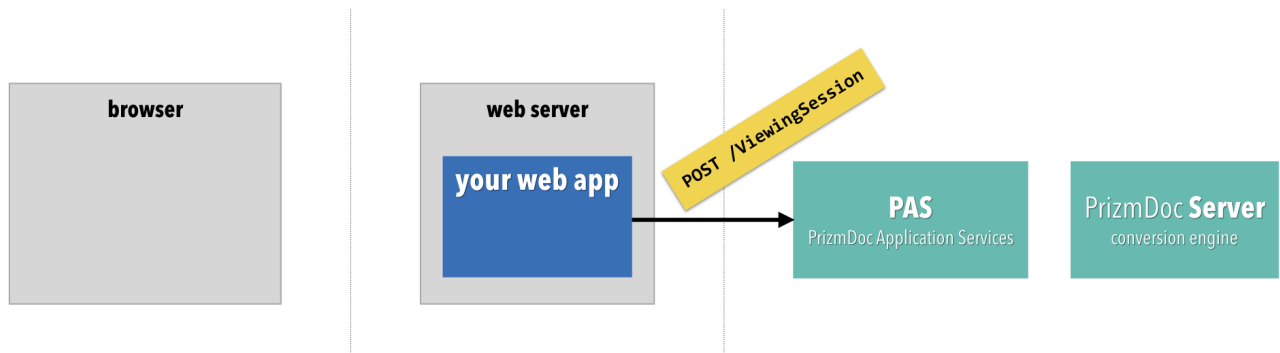


That error is the viewer's way of saying "Hey, you forgot to give me the id of my *viewing session*! I have no way to ask the backend for content."

Let's fix this.

Creating the Viewing Session

Before rendering HTML with the viewer and sending it to the browser, your web application must first send a POST request to PAS to create a new *viewing session*:



If you're using [PrizmDoc Cloud](#), the HTTP request to create a *viewing session* looks like this:

```
POST https://api.accusoft.com/prizmdoc/ViewingSession
Acs-Api-Key: YOUR_API_KEY
Content-Type: application/json

{
  "source": {
    "type": "upload",
    "displayName": "UNIQUE_NAME_OF_THE_DOCUMENT_TO_BE_VIEWED"
  }
}
```

NOTE: If you're using a self-hosted backend:

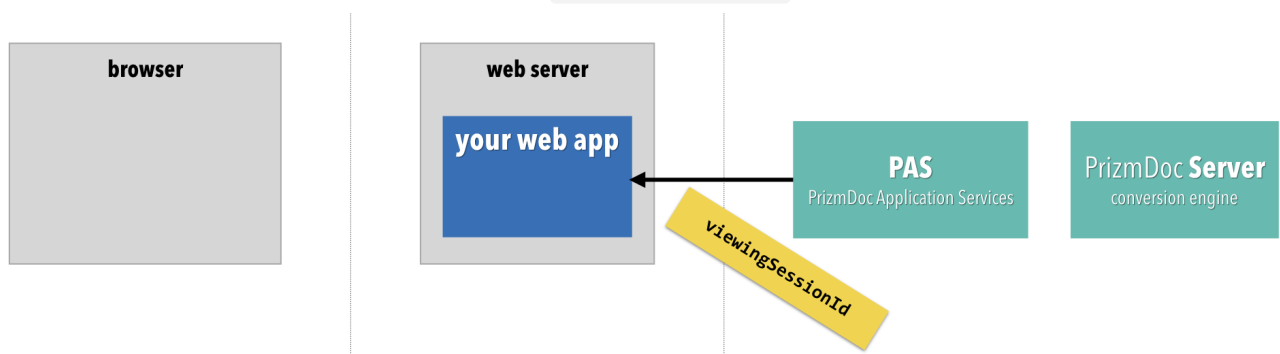
- Replace `https://api.accusoft.com/prizmdoc` with your actual PAS base URL (such as `http://localhost:3000`)
- Omit the `Acs-Api-Key` request header (needed for PrizmDoc Cloud)

Let's break this down:

- The `source` object tells PAS information about the original source document.
- `source.type` tells PAS how you intend to provide the source document. The value of "upload" means that you will be *uploading* the source document to PAS in a subsequent HTTP request, associating it with this *viewing session* (there are other options available, but "upload" is the one we recommend almost all of the time).
- Finally, `source.displayName` is a required option when `source.type` is set to "upload", and it must be a unique name for the document.

NOTE: To optimize backend performance, `source.displayName` must be unique for the binary contents of the document you will upload. You should NEVER use the same "displayName" value for two different source documents.

PAS will create a new *viewing session* and reply with its `viewingSessionId`:

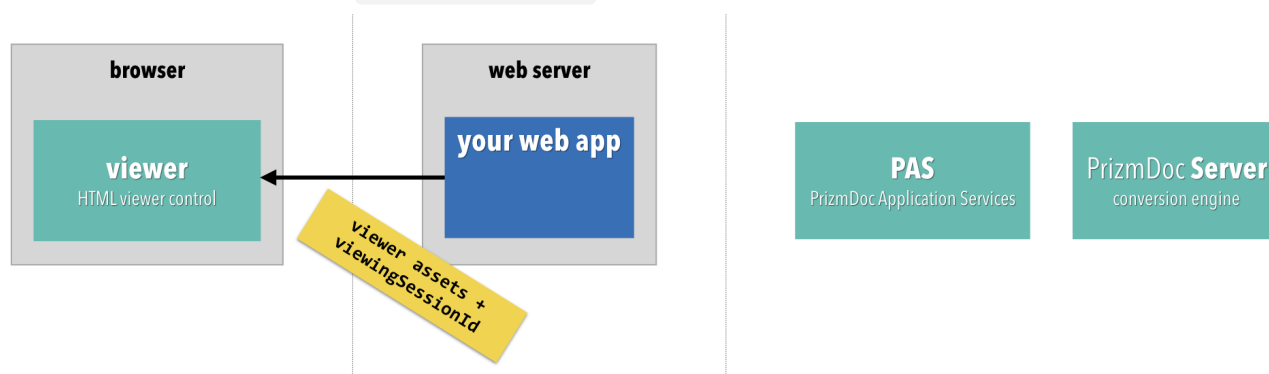


```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "viewingSessionId": "XYZ..."
}
```

Connecting the Viewer to the Viewing Session

At this point, we've created a new "empty" *viewing session* and we have its unique `viewingSessionId`. PAS is still waiting on us to upload the source document. However, before we do that, we can go ahead and render the HTML with a viewer, configuring it to use this new `viewingSessionId`:



We construct the viewer just as we did before in [part 1](#), but **this time we will provide a value for the required `documentID` property which the viewer was complaining about** (the viewer API calls it `documentID` and the PAS REST API calls it `viewingSessionId`, but they are the same thing):

```
<script type="text/javascript">
  $(function() {
    $('#viewerContainer').pccViewer({
      documentID: 'XYZ...',
      imageHandlerUrl: '/pas-proxy',
      viewerAssetsPath: 'viewer-assets',
      resourcePath: 'viewer-assets/img',
      language: viewerCustomizations.languages['en-US'],
      template: viewerCustomizations.template,
      icons: viewerCustomizations.icons,
      annotationsMode: "LayeredAnnotations"
    });
  });
</script>
```

Uploading the Source Document to Begin Document Conversion

Finally, after rendering the HTML and sending it to the browser, your web application needs to actually upload the source document to PAS, associating it with the *viewing session*.

If you're using [PrizmDoc Cloud](#), the HTTP request to upload the source document looks like this:

```
PUT https://api.accusoft.com/prizmdoc/ViewingSession/u{viewingSessionId}/SourceFile
Acs-API-Key: YOUR_PRIZMDOC_CLOUD_API_KEY
```

```
<<file bytes>>
```

NOTE: If you're using a self-hosted backend:

- Replace `https://api.accusoft.com/prizmdoc` with your actual PAS base URL (such as `http://localhost:3000`)
- Omit the `Acs-Api-Key` request header (needed for PrizmDoc Cloud)
- Add the required `Accusoft-Secret` header with the value of your PAS `secretKey`

If your document is accepted, PAS will reply right away with a simple `200 OK`:

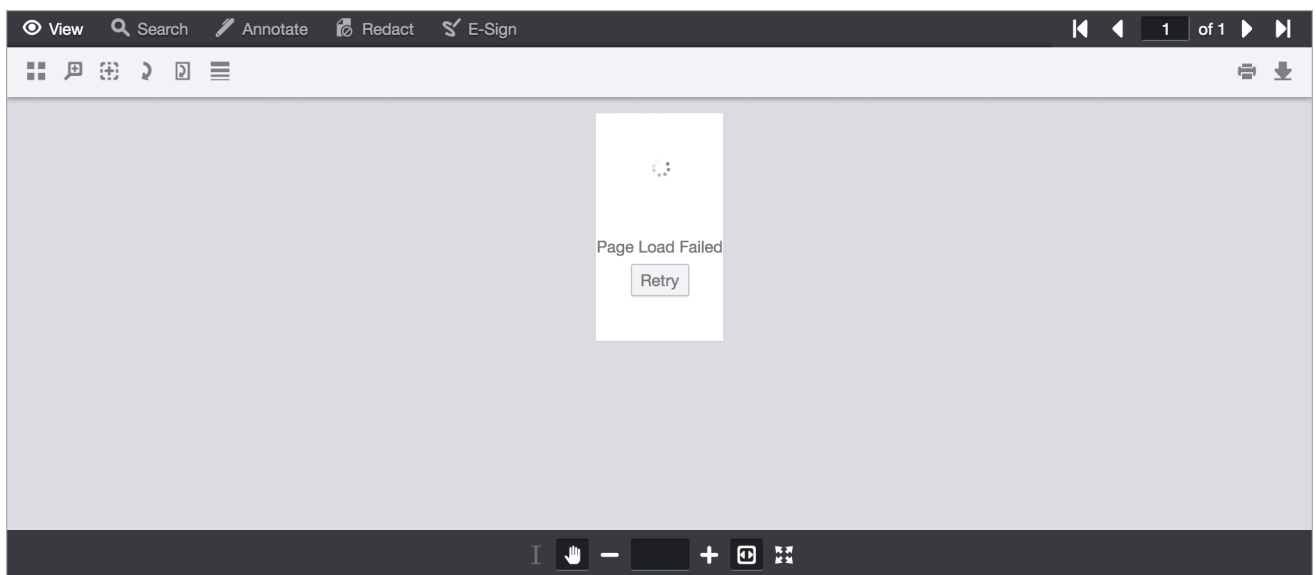
```
HTTP/1.1 200 OK
```

This just means that your document has been accepted and the conversion process has begun.

What We Have So Far

The viewer has a real `viewingSessionId`, and the backend is actually converting the document for viewing in the browser. And the viewer is now trying to ask the backend for converted document content.

If you run your web application now, the viewer should show a loading indicator. However, you'll also notice a "Page Load Failed" message and no content actually appears:



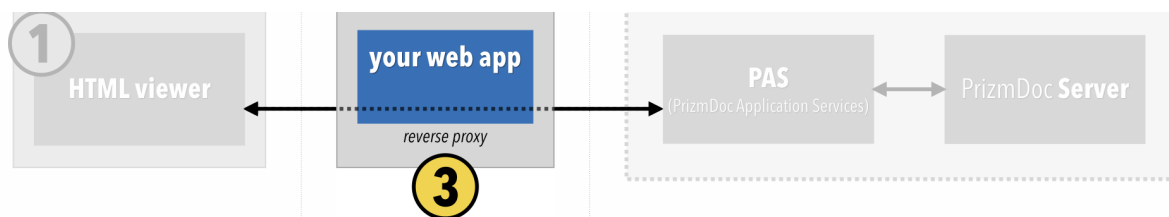
That's because the viewer's requests to the backend aren't actually getting through yet. We'll fix that next in [part 3](#).

3. Setting up a Reverse Proxy

Overview

Finally, we need to set up a simple reverse proxy so that the viewer can make requests *through* your web server to the backend:





By the end of this section, the viewer should actually display real document content.

Configuring the Viewer's Reverse Proxy Route

When we constructed the viewer, there was a special `imageHandlerUrl` property which we set to the value `/pas-proxy`:

```
<script type="text/javascript">
  $(function() {
    $('#viewerContainer').pccViewer({
      documentID: 'XYZ...',
      imageHandlerUrl: '/pas-proxy',
      viewerAssetsPath: 'viewer-assets',
      resourcePath: 'viewer-assets/img',
      language: viewerCustomizations.languages['en-US'],
      template: viewerCustomizations.template,
      icons: viewerCustomizations.icons,
      annotationsMode: "LayeredAnnotations"
    });
  });
</script>
```

`imageHandlerUrl` is just the base URL that the viewer should use when it makes GET requests to PAS for document content.

The viewer makes a variety of GET requests to PAS, such as:

- GET `/Page/q/0?DocumentID=uXYZ...&Scale=1`
- GET `/Document/q/Attributes?DocumentID=uXYZ...`
- GET `/Document/q/0-0/Text?DocumentID=uXYZ...`
- etc.

When the viewer makes those requests, it will use the `imageHandlerUrl` as the base URL for all of those requests. Since we set the value to `/pas-proxy`, the actual requests it makes will look like this:

- GET `/pas-proxy/Page/q/0?DocumentID=uXYZ...&Scale=1`
- GET `/pas-proxy/Document/q/Attributes?DocumentID=uXYZ...`
- GET `/pas-proxy/Document/q/0-0/Text?DocumentID=uXYZ...`
- etc.

Notice that these requests to `/pas-proxy/*` are still going to your web server. We still need to set up a reverse proxy for this `/pas-proxy/*` route to ensure that all such HTTP requests are forwarded to PAS.

The Reverse Proxy and Your PrizmDoc Cloud API Key

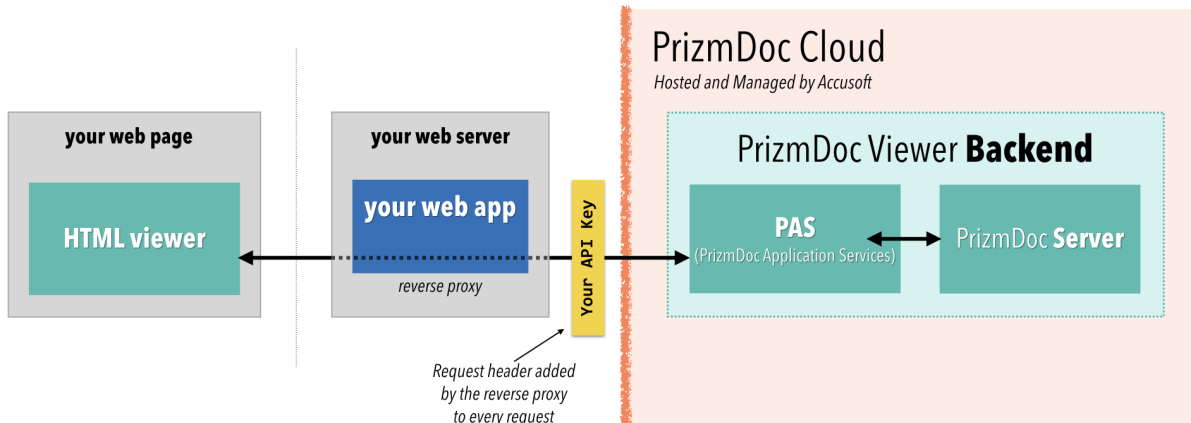
If you're using [PrizmDoc Cloud](#), your reverse proxy has an important responsibility: it must add your PrizmDoc Cloud API key to each HTTP request before sending it along to the backend.

Every HTTP request made to PrizmDoc Cloud *must* be authenticated with an API key provided in a request header named `Acs-Api-Key`.

However, you need to keep your API key private. It's what gives you access to everything within PrizmDoc Cloud, and it's what we use to determine how many billable transactions you've performed. To avoid people abusing your account and incurring unwanted charges, you should **never** send your API key to the browser. Instead, this responsibility should be handled by your reverse proxy.

So, the viewer will send GET requests to your web application. Your reverse proxy will accept the request, add the API key header,

and then forward the request along to PrizmDoc Cloud:



Setting Up a Reverse Proxy

There are lots of options for setting up a reverse proxy. Fundamentally, you need to decide whether to set up your reverse proxy 1) as part of your application code or 2) as part of an external web server, like [nginx](#) or [IIS](#). We offer examples of both. Pick one that most-closely matches what you need:

- [Setting Up a Reverse Proxy Within Your Web Application](#)
 - [node.js](#)
 - [ASP.NET](#)
 - [Java / Spring](#)
- [Setting Up a Reverse Proxy Outside of Your Web Application](#)
 - [nginx](#)
 - [IIS](#)

Setting Up a Reverse Proxy Within Your Web Application

node.js

If you're using [node.js](#) and [express](#), you can use a package like [http-proxy-middleware](#).

NOTE: For a complete example application, check out our [Hello PrizmDoc Viewer with node.js and HTML](#) sample on [GitHub](#).

Here is how you would set up a `/pas-proxy` route:

```
const express = require('express');
const proxy = require('http-proxy-middleware');

const app = express();

app.use(proxy('/pas-proxy', {
  pathRewrite: {
    '^/pas-proxy': '', // remove the /pas-proxy prefix when forwarding
    target: 'https://api.accusoft.com/prizmdoc', // PAS base URL
    changeOrigin: true, // necessary to convert from HTTP to HTTPS
    headers: {
      'Acs-Api-Key': 'YOUR_API_KEY' // for PrizmDoc Cloud
    }
  }
}));
```

And that's how you can configure your express application to proxy all `GET /pas-proxy/*` requests to PAS!

Once you've set that up, you can [skip ahead to the conclusion](#).

ASP.NET

If you're using ASP.NET, you can use [SharpReverseProxy](#), an OWIN middleware package available on [nuget](#).

NOTE: For a complete example application, check out our [Hello PrizmDoc Viewer with .NET and HTML sample on GitHub](#).

Within the `Configure` method of your `Startup.cs`, here is how you would set up a `/pas-proxy` route:

```
app.UseProxy(new List<ProxyRule> {
    new ProxyRule {
        Matcher = uri => uri.AbsolutePath.StartsWith("/pas-proxy/"),
        Modifier = (req, user) =>
        {
            // Create a corresponding request to the actual PAS host
            var match = Regex.Match(req.RequestUri.PathAndQuery, "/pas-proxy/(.+)");
            var path = match.Groups[1].Value;
            var pasBaseUri = new Uri("https://api.accusoft.com/prizmdoc/");
            req.RequestUri = new Uri(pasBaseUri, path);

            // For PrizmDoc Cloud
            req.Headers.Add("Acs-Api-Key", "YOUR_API_KEY");
        }
    }
}, result =>
{
    Logger.LogDebug($"Proxy: {result.ProxyStatus} Url: {result.OriginalUri} Time: {result.Elapsed}");
    if (result.ProxyStatus == ProxyStatus.Proxied)
    {
        Logger.LogDebug($"          New Url: {result.ProxiedUri.AbsoluteUri} Status: {result.HttpStatusCode}");
    }
});
```

And that's how you can configure your ASP.NET application to proxy all `GET /pas-proxy/*` requests to PAS.

Once you've set that up, you can [skip ahead to the conclusion](#).

Java / Spring

In a Java Spring application you can use [Netflix Zuul](#).

NOTE: For a complete example application, check out our [Hello PrizmDoc Viewer with Java and HTML sample on GitHub](#).

Your `application.properties` will define the information Zuul needs to set up a reverse proxy route for `/pas-proxy`:

```
prizmdoc.pas.baseUrl=https://api.accusoft.com/prizmdoc/
prizmdoc.cloud.apiKey=YOUR_API_KEY

# ===== Proxy all requests from /pas-proxy/* to PAS =====
ribbon.eureka.enabled=false
zuul.routes.pas-proxy.path=/pas-proxy/**
zuul.routes.pas-proxy.url=${prizmdoc.pas.baseUrl}
zuul.routes.pas-proxy.stripPrefix=true
```

Your main `Application.java` will look like this:

```
package myapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;
```

```
import org.springframework.context.annotation.Bean;
import sample.pasProxy.AddApiKeyRequestHeaderFilter;

@EnableZuulProxy
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

If you're using [PrizmDoc Cloud](#), you'll need to also make sure that your API key is added as a request header before requests are forwarded to the backend. You can do that by setting up a Zuul "pre" filter Spring component:

```
package myapp;

import javax.servlet.http.HttpServletRequest;
import com.netflix.zuul.context.RequestContext;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;

import com.netflix.zuul.ZuulFilter;

/**
 * Ensures that the PrizmDoc Cloud API key, if defined, is injected
 * as a request header before proxied requests are sent to PAS.
 *
 * Netflix Zuul will execute this "pre" filter before sending the request on to PAS, allowing
 * this class to inject the configured PrizmDoc Cloud API key.
 *
 * See https://spring.io/guides/gs/routing-and-filtering/
 */
@Component
public class AddApiKeyRequestHeaderFilter extends ZuulFilter {

    private static final Logger log =
        LoggerFactory.getLogger(AddApiKeyRequestHeaderFilter.class);

    @Value("${prizmdoc.cloud.apiKey:#{null}}")
    private String apiKey;

    @Override
    public String filterType() {
        return "pre";
    }

    @Override
    public int filterOrder() {
        return 0;
    }

    @Override
    public boolean shouldFilter() {
        return true;
    }

    @Override
    public Object run() {
        RequestContext ctx = RequestContext.getCurrentContext();
        HttpServletRequest request = ctx.getRequest();

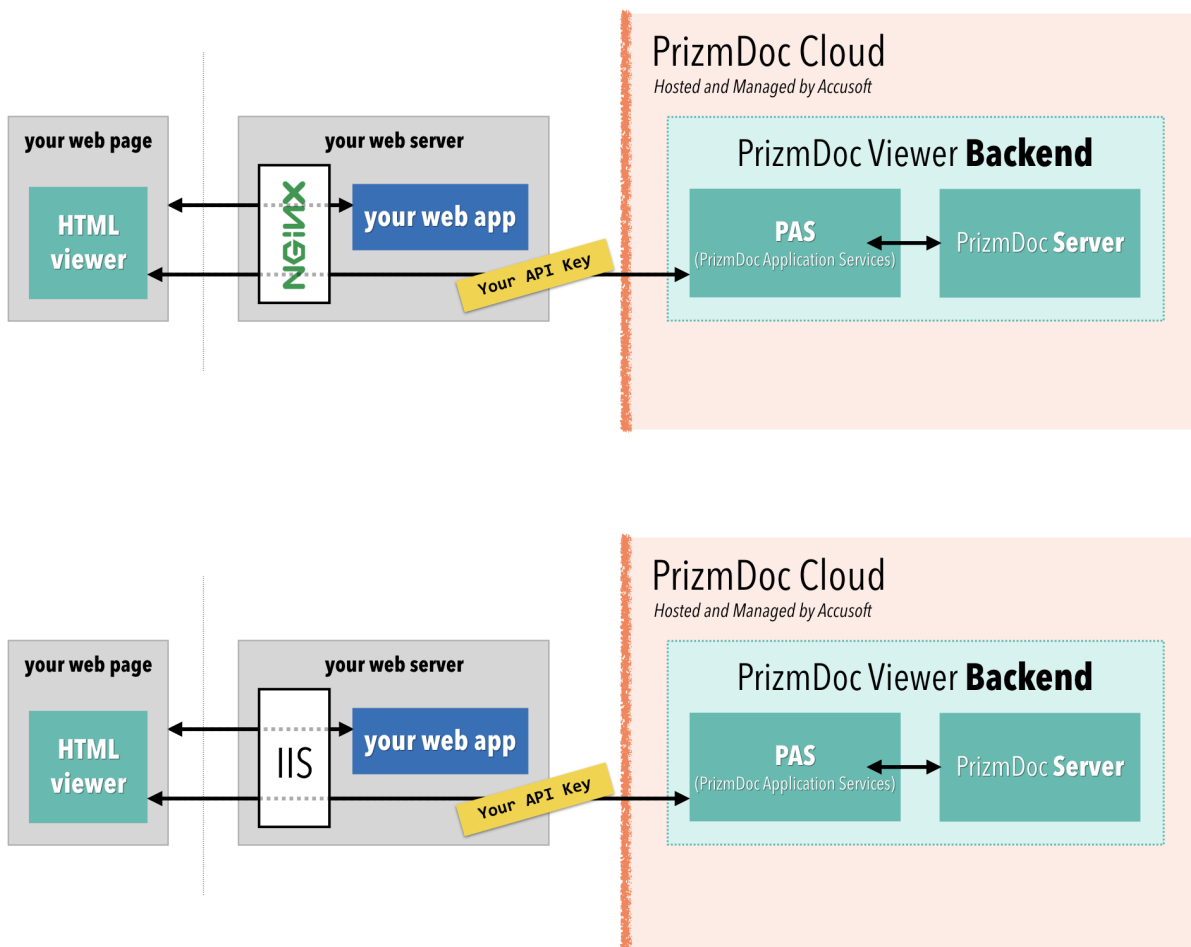
        if (apiKey != null) {
            ctx.addZuulRequestHeader("Acs-Api-Key", apiKey);
        }
    }
}
```

```
log.info("Proxying {} {}", request.getMethod(), request.getRequestURL());  
  
return null;  
}  
  
}
```

And that's how you can configure your Java Spring application to proxy all `GET /pas-proxy/*` requests to PAS. Once you've set that up, you can [skip ahead to the conclusion](#).

Setting Up a Reverse Proxy Outside of Your Web Application

In a production application, it's common to keep the reverse proxy concern outside of your application code, using something like nginx or IIS in front of your web application:



With this approach, all incoming traffic goes to nginx or IIS and then, based on the URL, gets routed either to your web application or to PAS.

Assuming you already have nginx or IIS in front of your web application, this section will explain how to configure an additional reverse proxy rule to route all `GET /pas-proxy/*` requests to PAS.

nginx

Within your nginx config file, you can easily add a `location` directive to a `server` defining a reverse proxy route to PAS:

```
location /pas-proxy/ {  
    # Limit to GET requests (which is all the viewer needs)
```

```
limit_except GET {
    deny all;
}

# For PrizmDoc Cloud:
proxy_set_header Acs-API-Key YOUR_API_KEY;

# PAS base URL
proxy_pass https://api.accusoft.com/prizmdoc/;
}
```

Note carefully the trailing slashes in the example above! They are important.

And that's how you configure nginx to proxy all `GET /pas-proxy/*` requests to PAS.

Once you've set that up, you can [skip ahead to the conclusion](#).

IIS

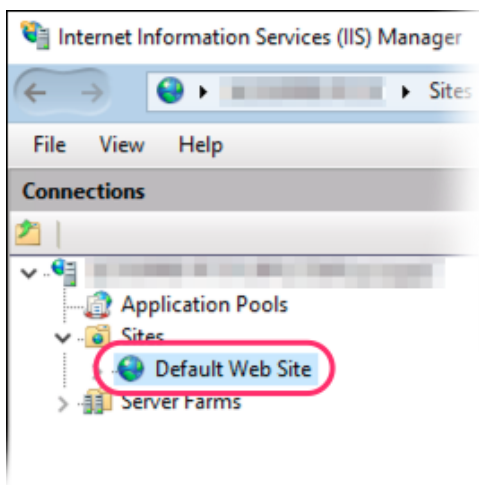
This section will walk you through setting up a reverse proxy route in IIS.

First, make sure you have these IIS extensions installed:

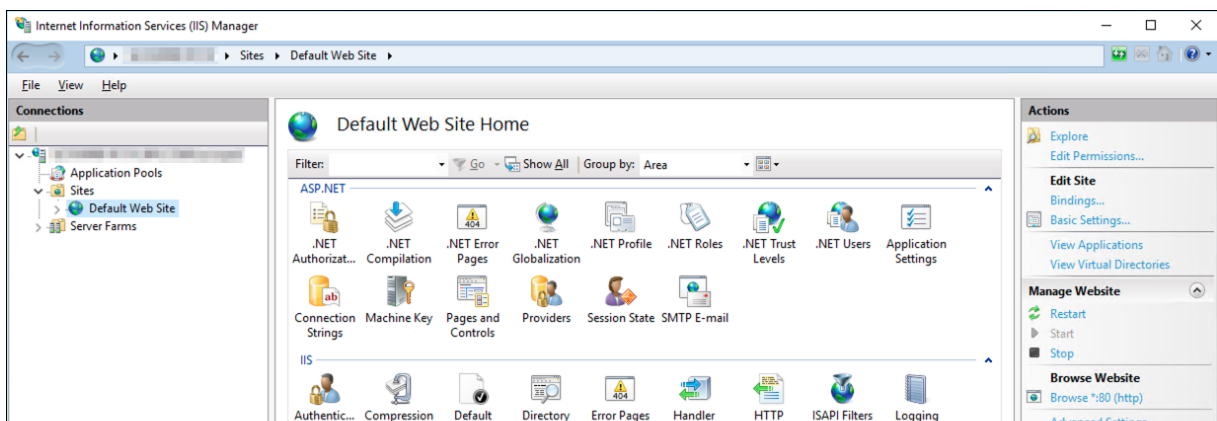
- [URL Rewrite](#)
- [Application Request Rerouting](#)

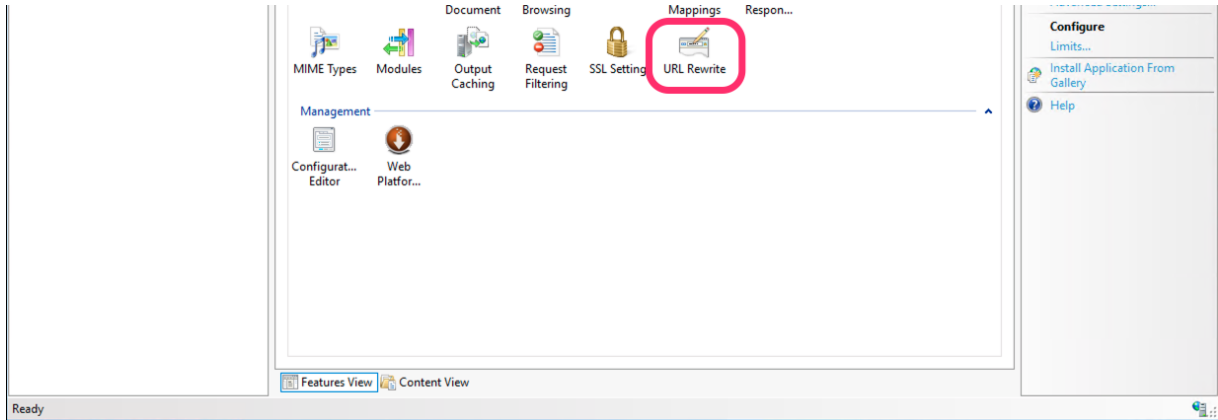
Next, we need to actually define the reverse proxy rule for your Site or Application in IIS. This sort of rule can only be applied at the Site or Application level. Let's walk through how to set this up:

1. Select the Site or Application you want to modify, such as **Default Web Site**:

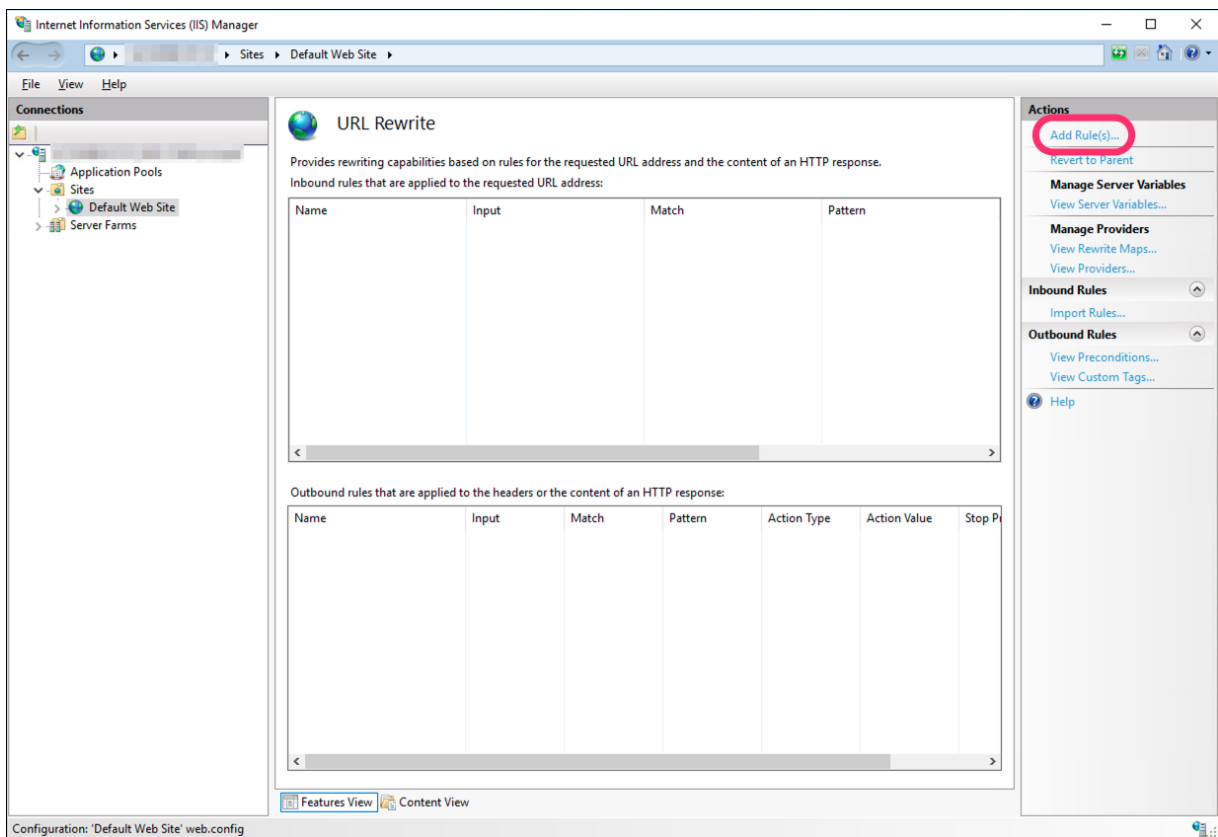


2. Open the **URL Rewrite** feature:

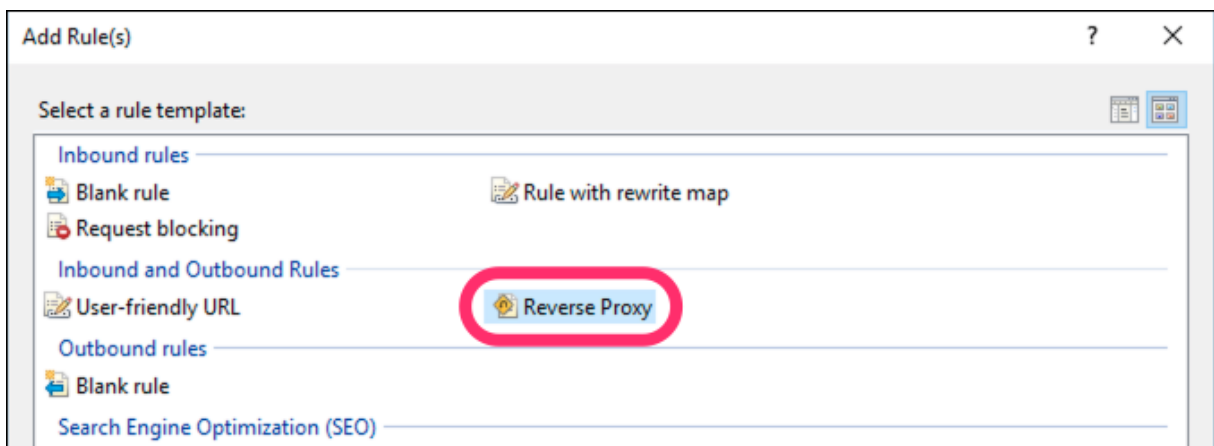


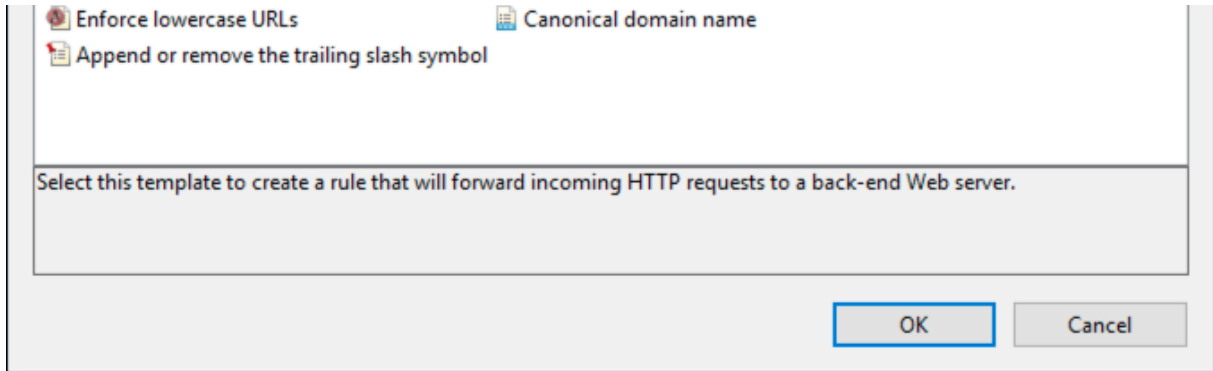


3. Click **Add Rule(s)...**



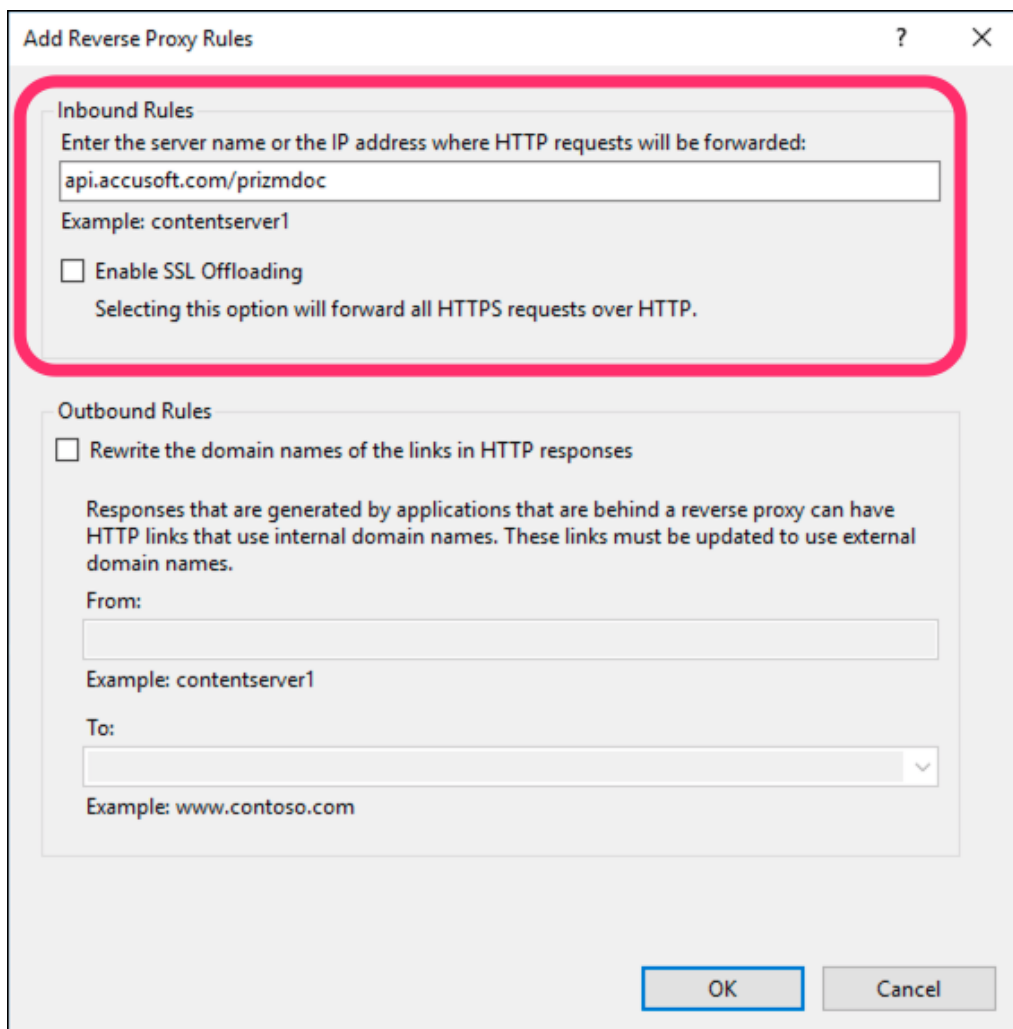
4. Choose the **Reverse Proxy** template:



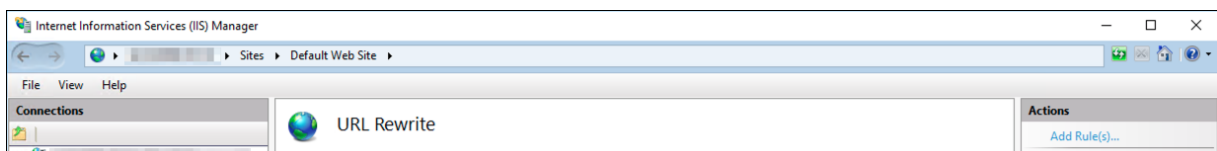


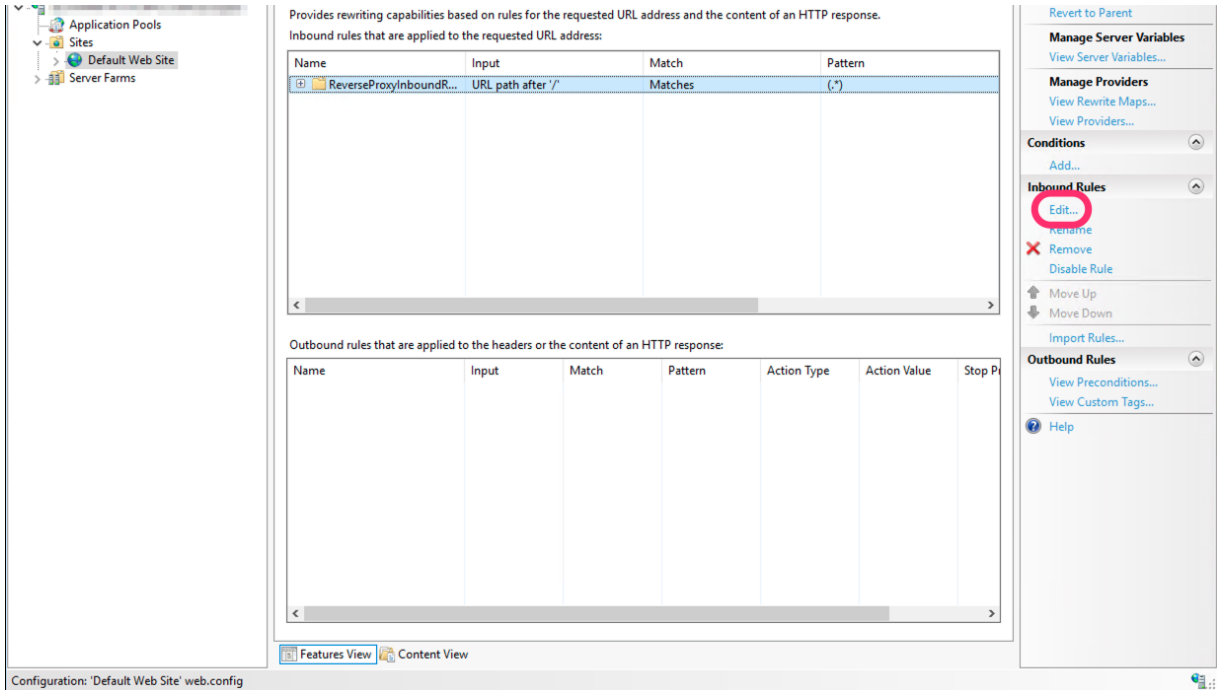
5. In the dialog:

- o Enter a **value** for your PAS host. If you're using PrizmDoc Cloud for your backend, use the value `api.accusoft.com/prizmdoc` for now.
- o Uncheck the **Enable SSL Offloading** checkbox.
- o Click **OK** to create the new rule.

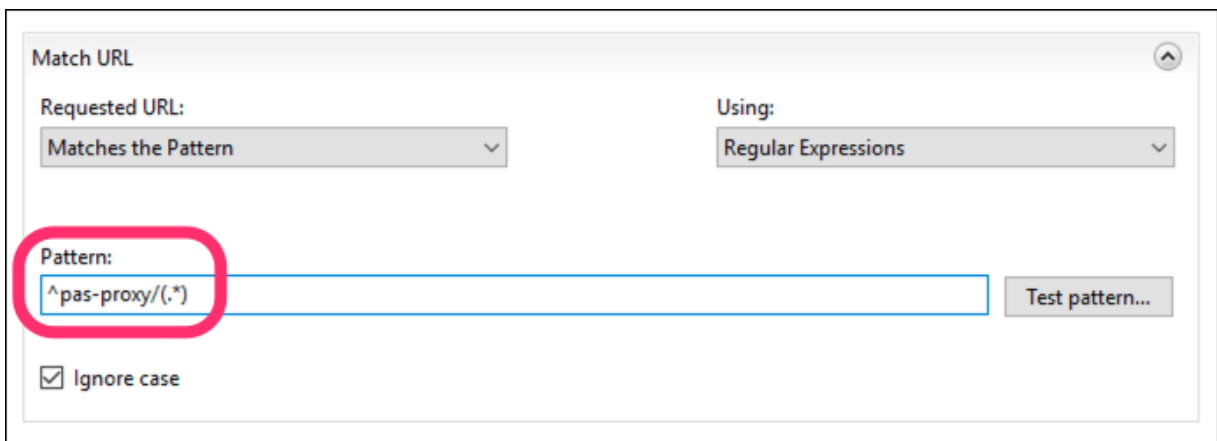


6. Now, click **Edit...** to further modify the rule you just created:

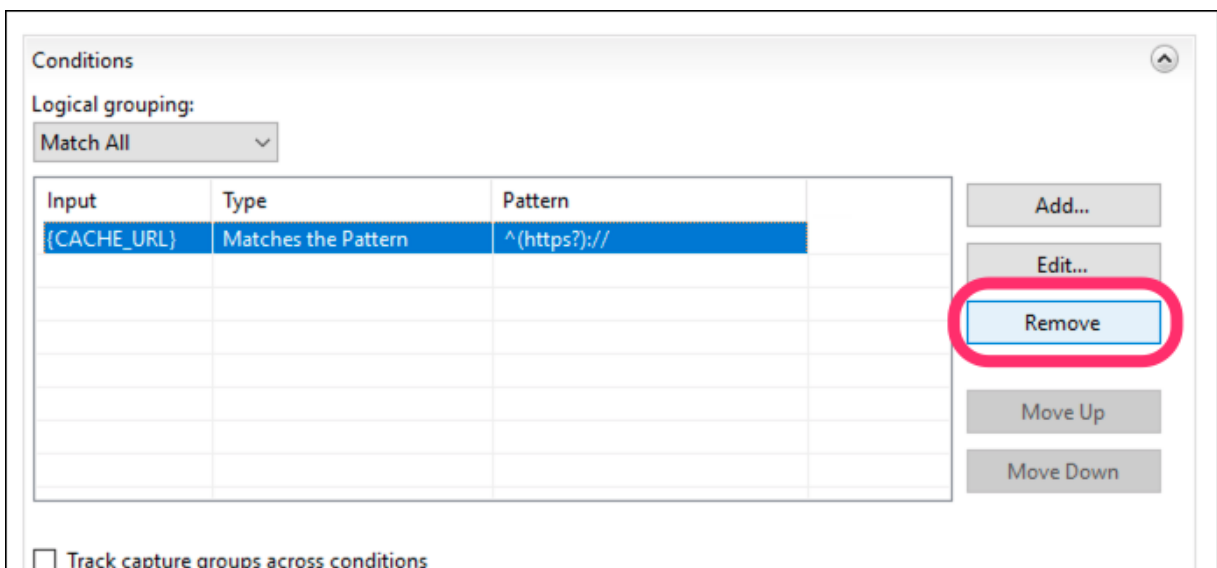




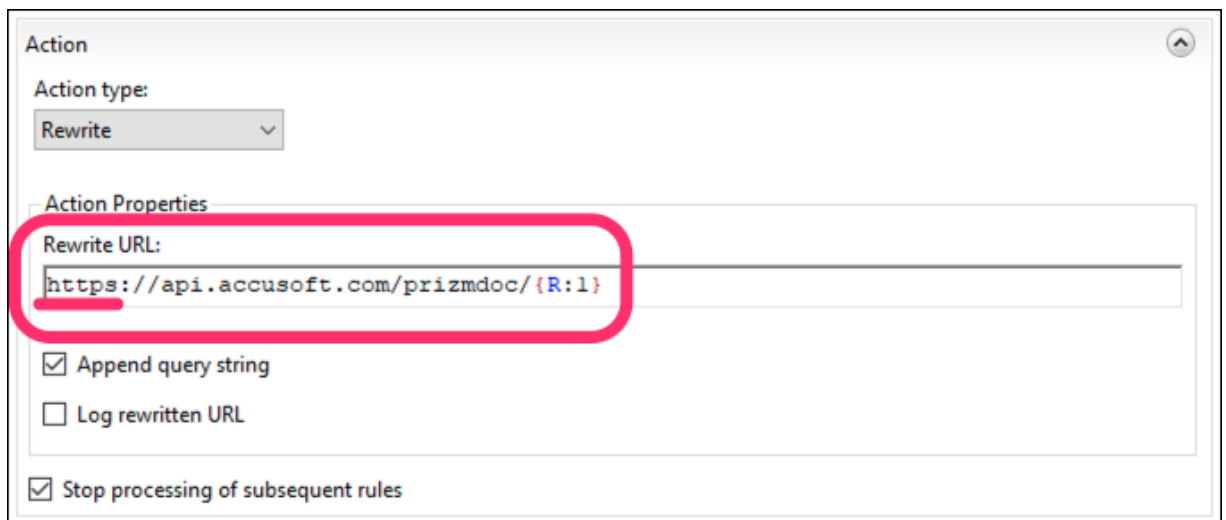
- Under **Match URL**, change the **Pattern** value to `^pas-proxy/(.*)`



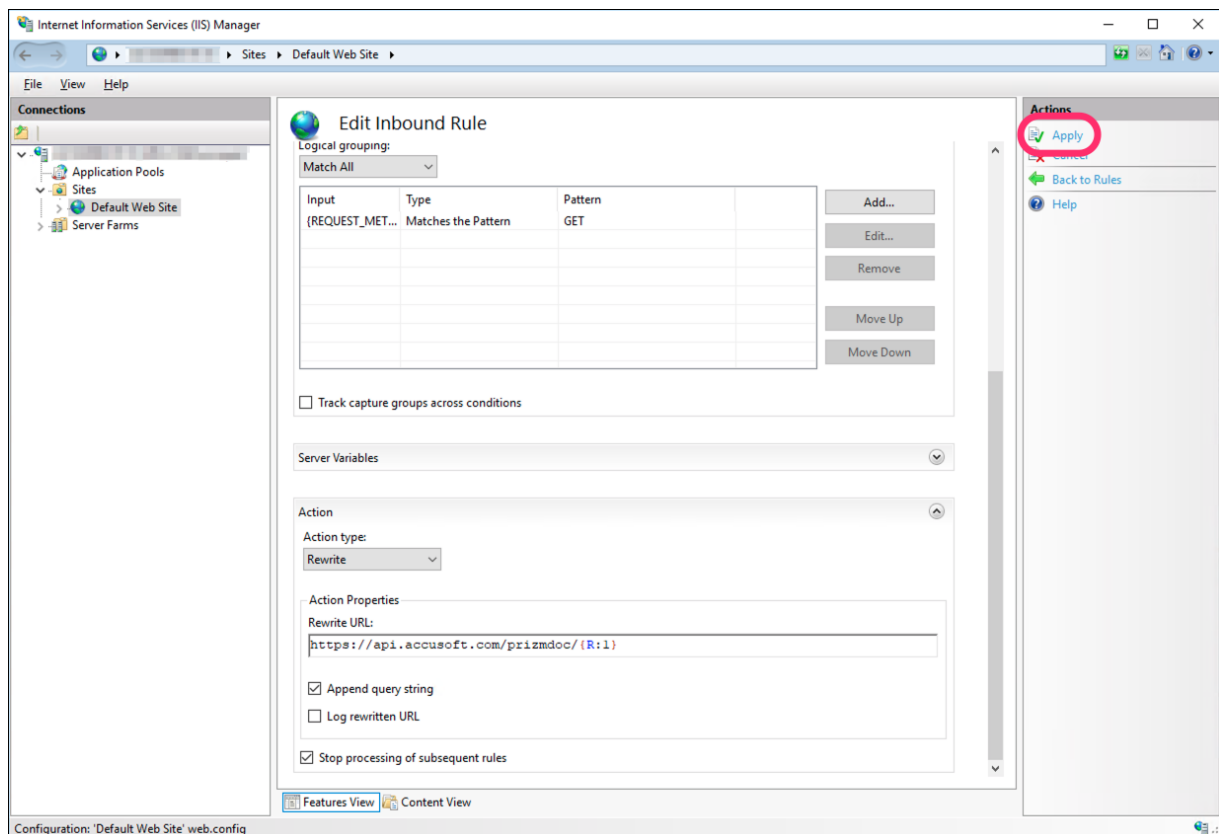
- Under **Conditions**, click **Remove** to delete the default `{CACHE_URL}` rule that was created:



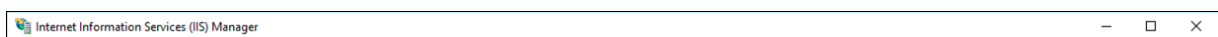
- At the bottom of the page, under **Action**, adjust the **Rewrite URL** to your actual PAS base URL, using `https` if possible.
 - The default **Rewrite URL** value begins with `{C:1}` to dynamically use `http` or `https` based on the original request. But, if your PAS instance is running on HTTPS, you should replace `{C:1}` with `https` to ensure that all requests to PAS are over HTTPS.
 - If you're using PrizmDoc Cloud for your backend, change the **Rewrite URL** to `https://api.accusoft.com/prizmdoc/{R:1}`:

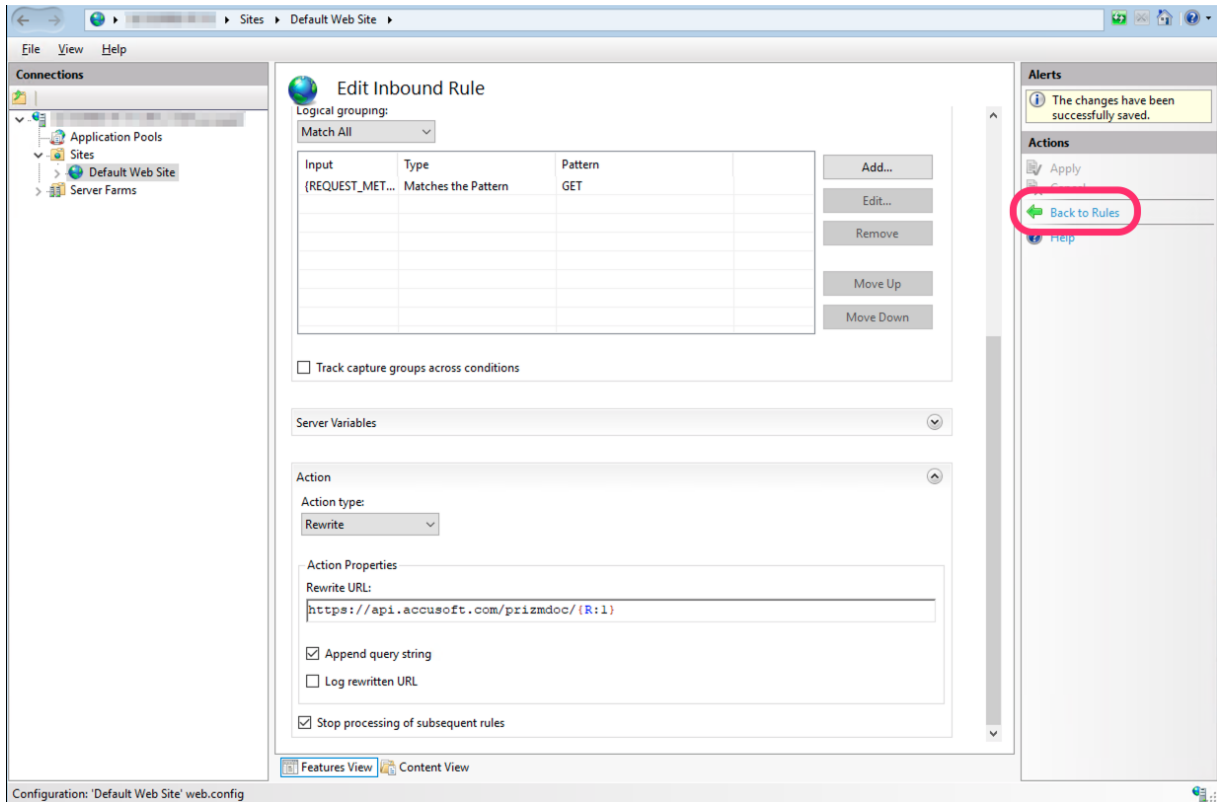


- Click **Apply** to save your changes to the rule:



- Click **Back to Rules**:



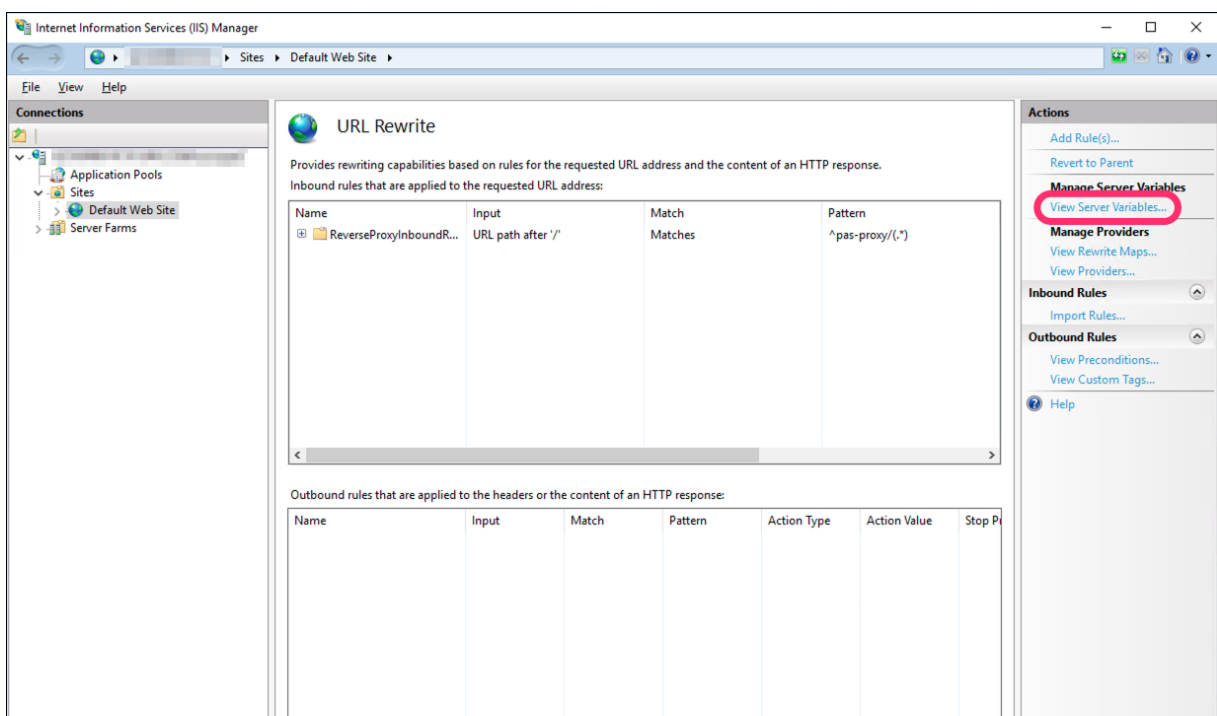


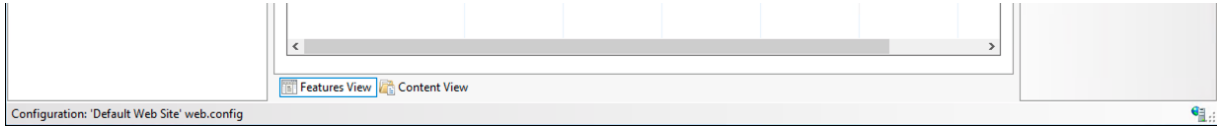
Finally, if you're using PrizmDoc Cloud for your backend, you need to ensure the reverse proxy adds your API key to every request before it sends it along to PAS. PrizmDoc Cloud requires that your API key be provided via an HTTP request header named `ACS_API_KEY`. With IIS, you can achieve this by setting an IIS *server variable* named `HTTP_ACS_API_KEY` to your actual API key value (the URL Rewrite module will add a request header for any server variable beginning with `HTTP_`, basing the header name on the rest of the server variable name; see the [URL Rewrite module docs](#) for more information).

Setting up this server variable is a two-step process.

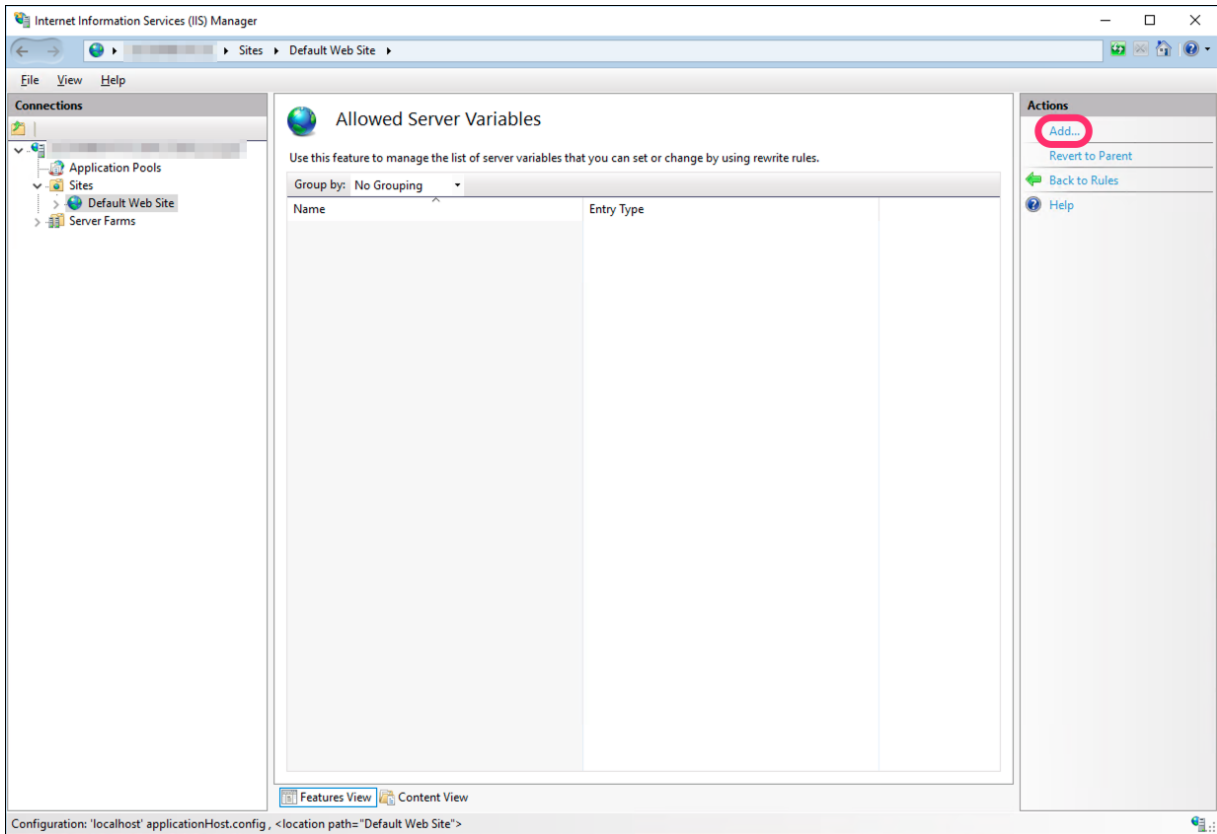
First, we need to define the `HTTP_ACS_API_KEY` server variable so we can actually use it.

1. On the URL Rewrite module screen, click **View Server Variables...**



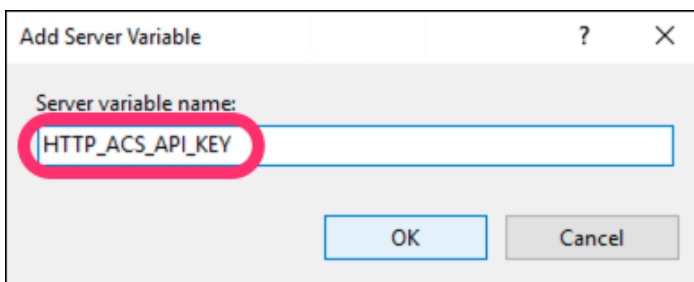


2. Click **Add...**

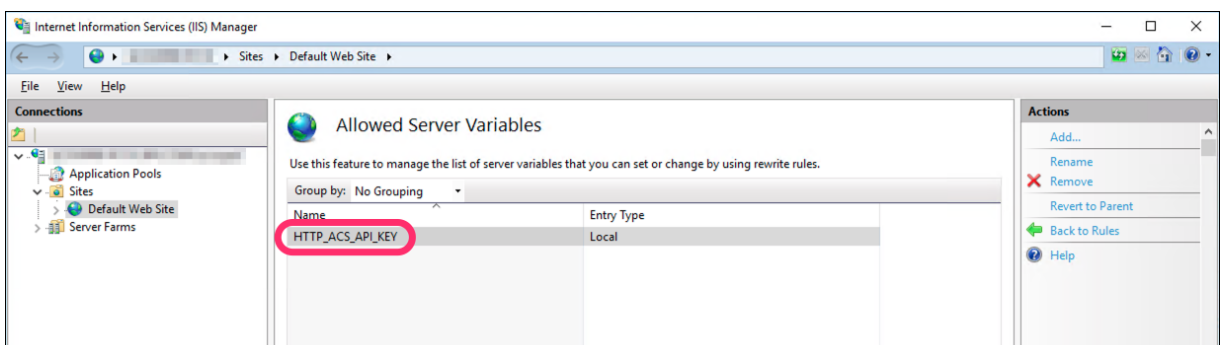


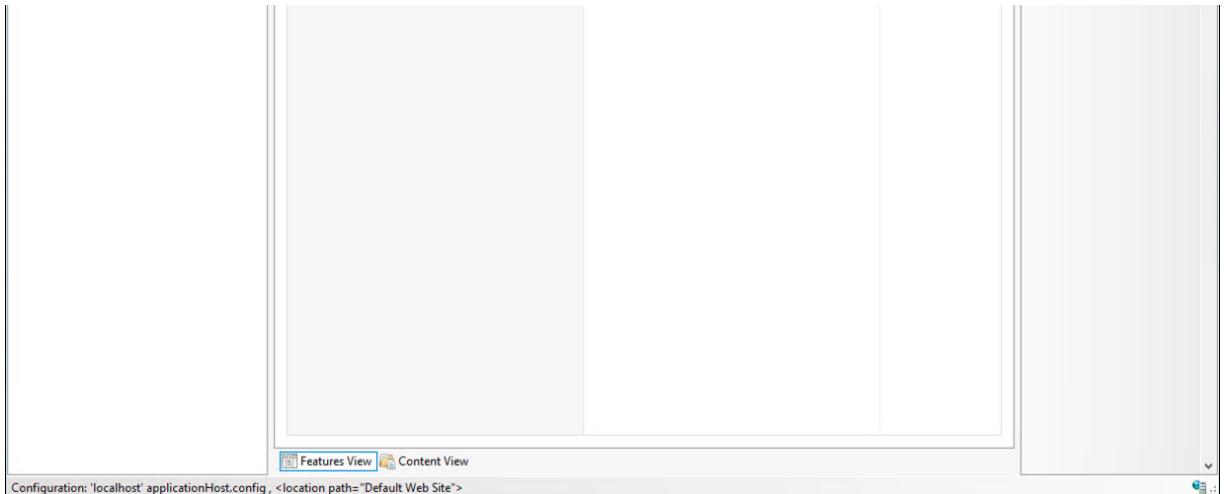
3. In the dialog:

- Set the **Server variable name** to `HTTP_ACS_API_KEY`:



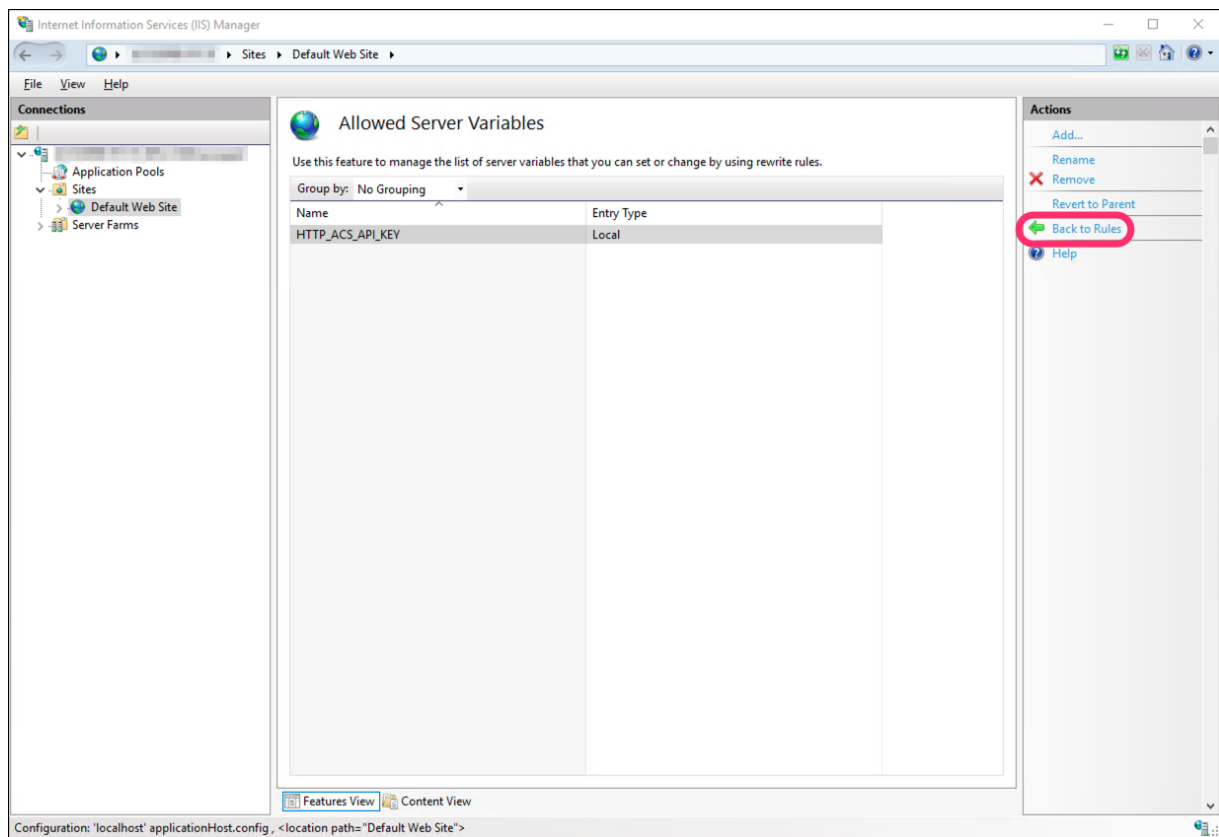
- Click **OK**. The **Allowed Server Variables** screen should now look like this:



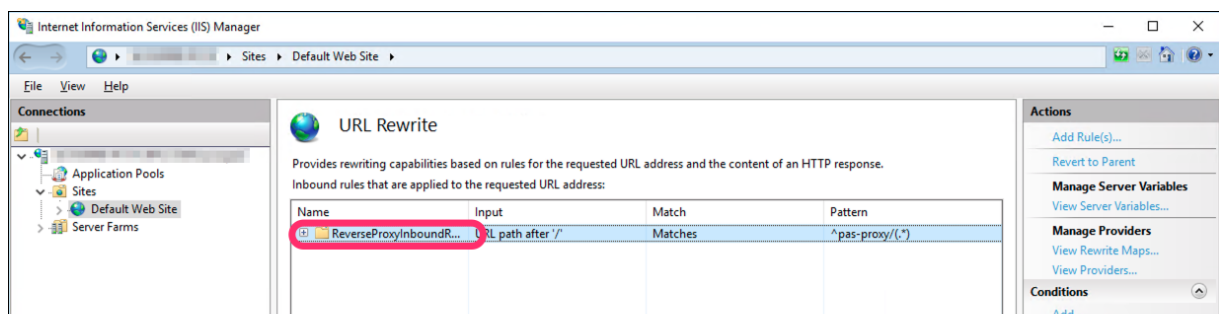


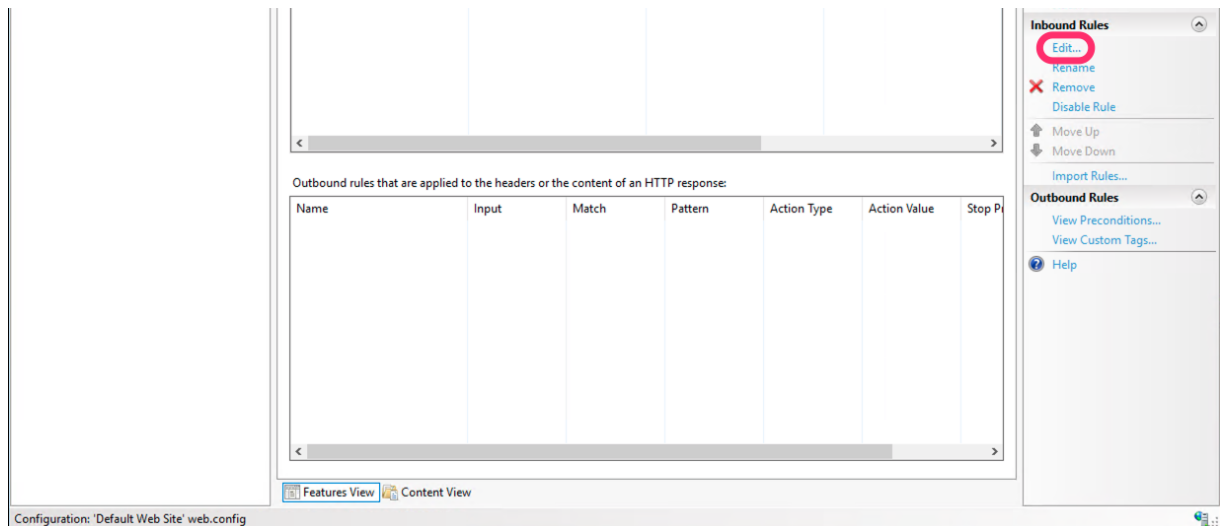
Now that we've allowed this variable, we need to actually use it in our reverse proxy.

1. Click **Back to Rules** to return to the list of URL Rewrite rules:

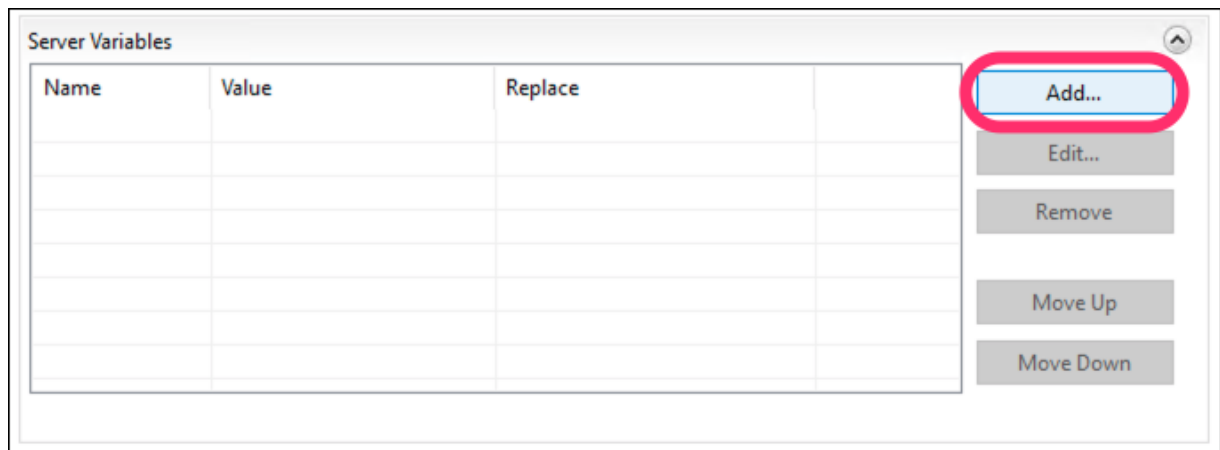


2. Select the reverse proxy rule we created earlier, then click **Edit...**



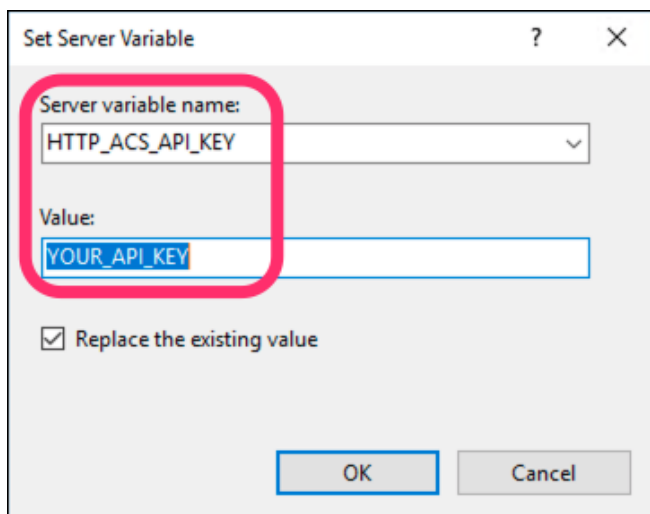


3. Expand **Server Variables** and click **Add...**



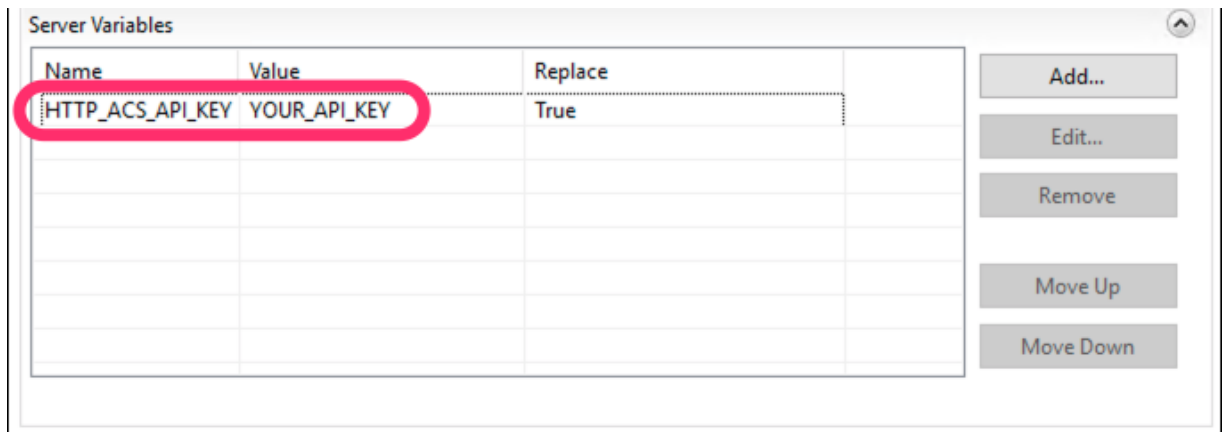
4. In the dialog:

- Set **Server variable name** to `HTTP_ACS_API_KEY`
- Set **Value** to your actual API key

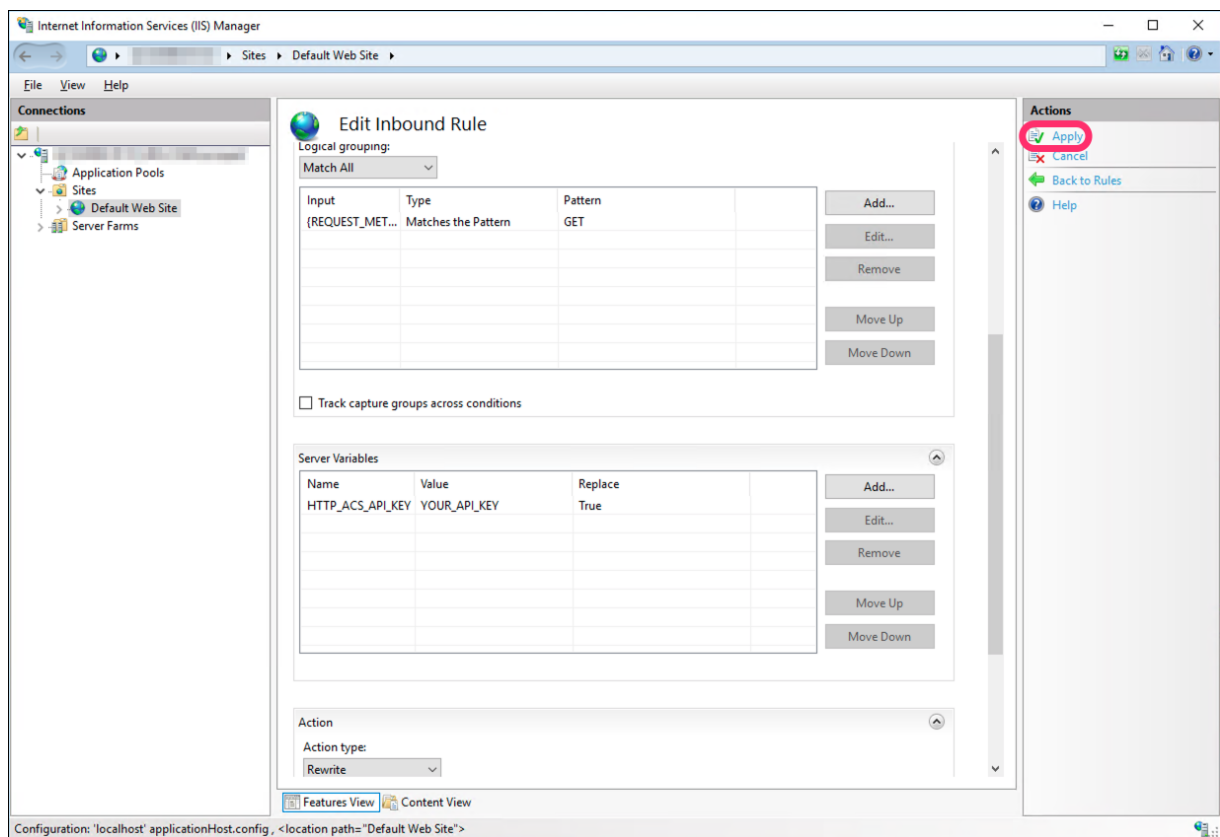


- Click **OK** to create this new server variable, which will add an `acs-api-key` request header to every request before forwarding it on to PAS. Your **Server Variables** list should now look like this:





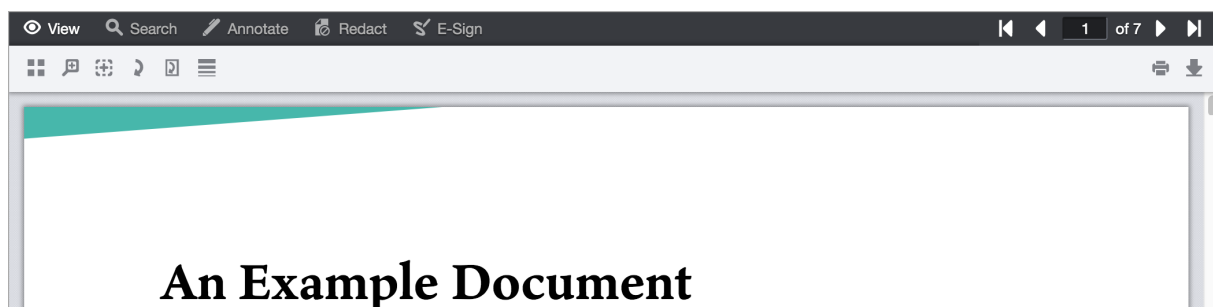
5. Click **Apply** to save your changes to the rule:



That's it! You've now configured your IIS site or application to proxy all `GET /pas-proxy/*` requests to PAS.

What We Have Now

If you've configured everything correctly, your web application should now display a viewer with actual document content:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque gravida euismod vehicula. Quisque semper commodo eros molestie condimentum. Praesent sagittis velit elit, eu porta metus tempus eget. Mauris ex

Cras at dignissim risus. Sed viverra ex eget erat volutpat vestibulum. Morbi at fermentum sem. Aenean luctus lobortis nisi, in vulputate nisi sagittis et. Maecenas rhoncus et leo non placerat. Suspendisse eget ultrices nibh, quis



Congratulations, you've integrated PrizmDoc Viewer into your application!

Developer Guide

Introduction

PrizmDoc Viewer was designed with ultimate flexibility for developers. With a completely customizable user interface and a powerful set of APIs for PrizmDoc Server, we provide numerous options to ensure that PrizmDoc Viewer meets your needs.

- [Customizing the Viewer](#)
- [Using the Viewer API](#)
- [The Two Backend Tiers](#)

Customizing the Viewer

There are several options to customize the Viewer based on your use case.

Using the Viewer Out-of-the-Box

The Viewer is designed to work out-of-the-box with only a few lines of integration code to write on your web tier. You may be able to integrate the Viewer into your web application with little or no customization needed.

Customizing through Configuration

If you need to make minor modifications to the Viewer, customizing through configuration is the simplest option. You can customize the Viewer by modifying the [uiElements](#) to control which tabs are displayed, as well as several other useful customizations.

For example, you may want to disable your users' ability to print and download documents. Initializing the Viewer with pre-defined search terms and localization can also be handled via client configuration. You can also control how pages are displayed for different sized documents through configuration. Modifying through configuration options means that you don't have to change the actual viewer code, minimizing integration time needed with future PrizmDoc Viewer releases.

Customizing the User Interface

The Viewer is designed using an open markup approach; all of the HTML and .css is open and customizable. This allows you to treat the Viewer either as an out-of-box product fully supported by Accusoft or as sample code. By taking the sample code approach, you can start with our complete Viewer and modify the Viewer code as needed, from minor tweaks to a complete re-design of the interface. Of course, with this approach, you will incur some overhead when merging customizations with future versions of [PrizmDoc Viewer](#).

The Viewer markup is made up of a number of HTML "template" files. The [template files](#) help segment the UI components and make it easier for you to focus on areas of the Viewer that you need to modify. Simple customizations such as rearranging, removing, or renaming tabs can be done very quickly by modifying the main template (**viewerTemplate.html**). From there, you can add, remove, or change anything in the Viewer UI, including designing a completely custom interface using the Viewer API. Additionally, we expose an unminified, unobfuscated JavaScript library that allows you to edit the business logic and behavior of the Viewer (see [Developer Guide > Customizing the Viewer > Modifying viewer.js](#)).

Using the Viewer API

The Viewer API permits programmatic control over the Viewer. Most API functionality is exposed by the ViewerControl - the core component of the Viewer. The Viewer UI/chrome builds off of the API members of the ViewerControl.

It is required to use the Viewer API for:

- Modifying the behavior of the Viewer (beyond simple configuration)
- Augmenting the behavior of the Viewer

- Building custom Viewer menus

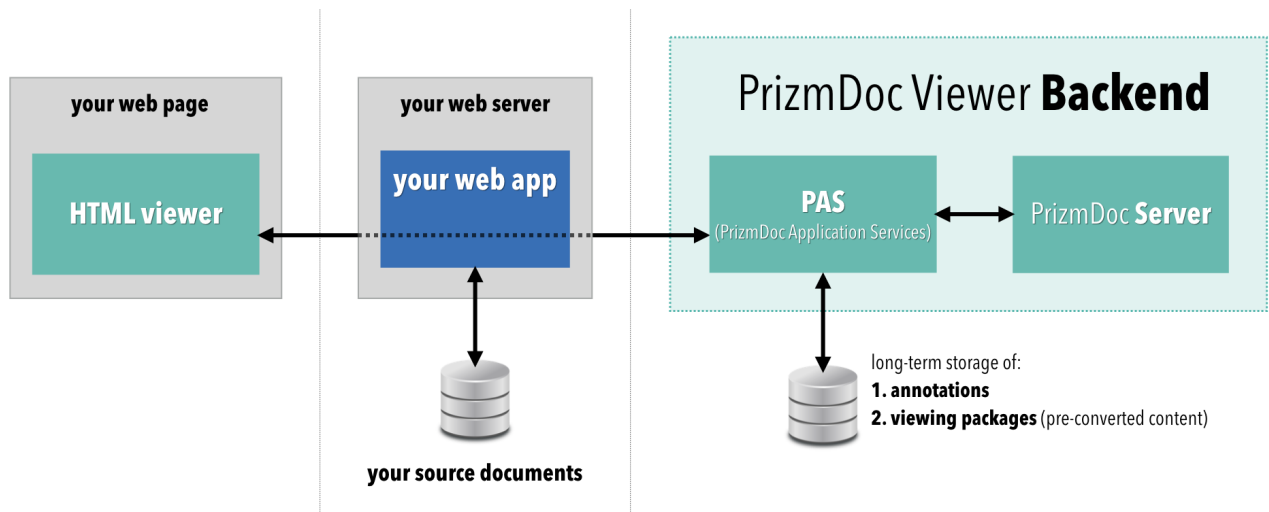
The Viewer API is *not* required for:

- Customizing the Viewer's layout or style
- Adding or removing tabs
- Moving or removing buttons and other inputs

For more information about the Viewer API, refer to the [API Reference](#) documentation.

The Two Backend Tiers

The backend is made up of two tiers, **PAS** (PrizmDoc Application Services) and **PrizmDoc Server**:



PAS and PrizmDoc Server are independent. Each runs on its own host or port, and each has its own REST API.

- **PrizmDoc Server** (on the far right) is the technical heart of the product, the actual engine that converts pages of a document to SVG. It is compute intensive and has no permanent storage.
- **PAS** does not do any conversion work. Instead, it is a layer in front of PrizmDoc Server which is responsible for other viewing concerns, such as saving and loading of annotations or long-term caching of pre-converted content. Like your web application, PAS has privileged access to storage that you own (like a file system or database).

For viewing functionality, your web application should only need to make REST API calls to PAS. PAS will make calls to PrizmDoc Server on your behalf to ensure the conversion work is actually done.

For automated document processing, your web application can use the powerful PrizmDoc Server REST API directly. For example, you can leverage the PrizmDoc Server REST API to convert files, combine files, burn markup or redactions into a file, and more.

PAS

For more information about **PAS**, see:

- The [PrizmDoc Application Services](#) section in this developer guide.
- The [PAS REST API reference](#).

PrizmDoc Server

For more information about **PrizmDoc Server**, see:

- The [PrizmDoc Server](#) section in this developer guide.
- The [PrizmDoc Server REST API reference](#).

Implement our Top Features

Introduction

This topic is a quick reference for you to review our top features and decide which ones you want to implement:

- **Customize the Viewer** - There are several ways to customize the Viewer, from quick integrations with minimal configuration to complete control over the Viewer API. You can create a chrome-less viewer or create your own custom controls including page navigation, zooming, thumbnails, searching, and printing. The following levels of customization are available:
 - **Level 1 - Out-of-the-Box** - By adding the [jQuery plugin](#) to your web application with minimal code, you can use the full features of the Viewer with no customization necessary.
 - **Level 2 - Customize Using Configuration Parameters** - The Viewer has a number of configuration options that allow you to control basic functions like tab display and localization just by launching the Viewer using [Initialization Parameters](#) and [uiElements](#). For many developers, the flexibility provided here will be sufficient.
 - **Level 3 - Simple Interface Customization** - By having an open markup and UI template design, minor customization such as moving, hiding, and renaming UI elements can be done by modifying [HTML templates](#). You can also inject your own code to add additional functionality and workflow.
 - **Level 4 - Advanced Customization** - With a complete viewer API, you may opt to do advanced customization, including building your own viewer from the ground up. See the [Developer Guide](#) for more details.
- **Annotate Documents** - A complete set of annotation tools including text commenting, image stamps, highlights, hyperlink, and polyline annotations and more.
- **Redact Documents** - Auto-redact features that are perfect for eDiscovery, research, and other collaborative applications where security is a top priority.
- **View Large Documents** - Minimize the load time and optimize performance for viewing and searching large documents. Server-side search reduces memory load in the Viewer, increasing performance of large document search results.
- **Pre-Convert Documents** - Use the Pre-conversion API in PrizmDoc Application Services (PAS) to experience nearly instantaneous viewing of documents.
- **Natively Render Microsoft Office Documents** - A valuable Microsoft Office Conversion (MSO) feature that provides true native viewing of Word, Excel, and PowerPoint documents.
- **Detect PDF AcroForm or Raster File** - The RESTful API quickly and accurately auto-detects form fields to convert existing documents into fillable forms that are simple to fill out and sign. Convert text, checkbox, and signature fields on AcroForm documents or automatically detect text fields from scanned TIFF or bitmap documents, creating interactive fillable form fields.
- **Document Comparison** - Document comparison is the process of cross-checking new versions of a document against previous versions so that you can see the changes. These changes could include formatting modifications such as font or spacing changes, grammatical changes, or the addition or omission of words, sentences, clauses or paragraphs. For more details on how to implement Document Comparison, refer to the following topics:
 - [Work with Document Comparison Programmatically](#)
 - [Perform Document Comparison](#)

Work Effectively with Large Documents

Introduction

This section gives high-level guidelines to achieve fast end-user interaction with source documents that contain hundreds or even thousands of pages.

Use Viewing Packages to Pre-Convert Content Whenever Possible

The most important thing you can do to make large documents load quickly in the browser is to make sure the document content has already been converted for viewing in the browser before an end user starts to view it. This is especially true for Microsoft Office documents.

If you are using PrizmDoc Application Services (PAS), you can take advantage of our [Viewing Packages](#) feature to comprehensively pre-convert an entire document for fast viewing in the browser. Once created, a viewing package persists until you explicitly delete it, and it allows PAS to simply return static content for any page of a document, even if the document has thousands of pages.

If you are not yet taking advantage of PAS (that is, you are communicating directly to the back-end PrizmDoc Server to create your viewing sessions), consider adding [PAS](#) to your environment to take advantage of the [Viewing Packages](#) feature.

If you are not familiar with how the browser, your web tier, PAS, and the back-end PrizmDoc Server work together, refer to the [PrizmDoc Overview](#).

Use Server-Side Search to View Large Documents

Ideally, the Viewer would perform a client-side search whenever possible and a server-side search whenever necessary. In reality, we make an educated guess based on page count. By default, our Viewer will perform a client-side search if a document contains no more than 80 pages; otherwise, the Viewer will offload the search work to the server. For many kinds of documents, this arbitrary 80-page threshold works fine. However, if you are using documents of 80 pages or fewer with a substantial amount of text, or if your end user's browser is particularly memory constrained, you may find that this default is not aggressive enough in offloading search work to the server.

When constructing a viewer control, you can use the [ViewerControlOptions](#) `searchMethodPageCountThreshold` property to adjust the maximum number of pages a document can have before the Viewer switches to server-side search. Additionally, you can use the `searchMethodType` property to force the Viewer to only use server-side search (or only use client-side search).

The [Viewing Sessions](#) `serverSideSearch` property determines whether server-side text searching will be available for a document and is the default value.

Viewer

This section contains the following information:

- [Architecture & Design](#)
- [Integrate PrizmDoc Viewer Releases with Your Code](#)
- [Configure the Viewer](#)
- [How to Customize the Viewer](#)
- [Modify viewer.js](#)

Architecture & Design

Introduction

The Viewer offers the following features out of the box:

- A responsive UI

- A jQuery plugin for embedding the full Viewer
- Configuration options
- A customizable UI
- An API
- Reusable core component

Responsive UI

The Viewer's responsive UI is designed for phone, tablet, and desktop users. A single UI implementation adapts to the viewport size of the device or element in which it is embedded.

jQuery Plugin

A jQuery plugin is used to embed the full-featured, responsive Viewer on the page.

Example

```
$("#myDiv").pccViewer(pluginOptions);
```

Configuration

The Viewer UI and behavior can be configured when the Viewer is embedded, using JavaScript parameters.

Example

```
var pluginOptions = {  
  documentID: "1234abcd",  
  encryption: false,  
  viewMode: "EqualWidthPages"  
};
```

Configurable options include:

- Disabling tabs
- DRM features
- Localization
- Rendering options
- Encryption
- Default tool settings
- Pre-defined search

Customizable UI

If the Viewer needs to be customized more than the configuration options allow, all of the UI code is open-source and can be modified to suit your customization needs. The open-source Viewer code is separated into CSS files, template HTML files, and JavaScript. The code leverages custom HTML attributes and Underscore.js' templating system in order to maintain the separation of concerns.

API

The Viewer API offers complete control over the Viewer. The API allows callers to augment, customize, or automate the end user's experience with the Viewer. The API functionality covers:

- Creating and destroying the Viewer
- Events
- Page navigation
- Zooming and fitting content
- Mark (annotation and redaction) CRUD
- Markup saving and loading
- Customizing mouse tools
- Searching document text
- Printing
- Getting page and document attributes

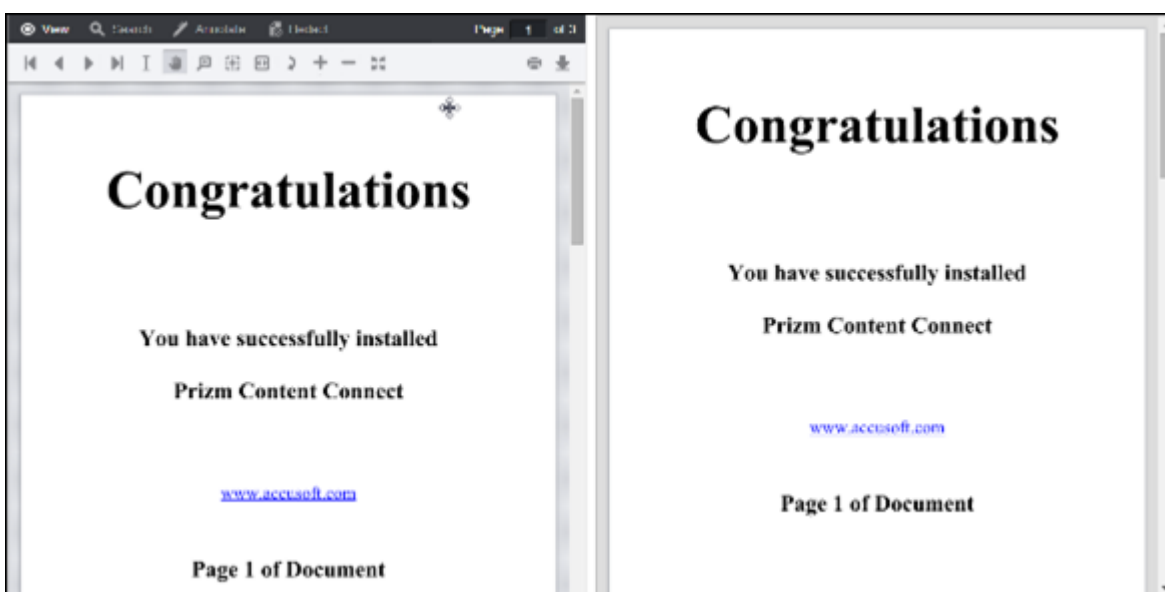
Example

```
var api = $("#myDiv").pccViewer(pluginOptions).viewerControl;  
api.on("PageCountReady", function() {  
    api.changeToLastPage();  
});
```

Reusable Core Component

The core component used by the Viewer for rendering the document is the ViewerControl. The ViewerControl is a component that can be used independent of the full Viewer; it can be directly embedded into a page and used for building a fully custom UI:

- The only UI of the ViewerControl is the page list, which allows scrolling through a document.
- The ViewerControl exposes the API for programmatic control.



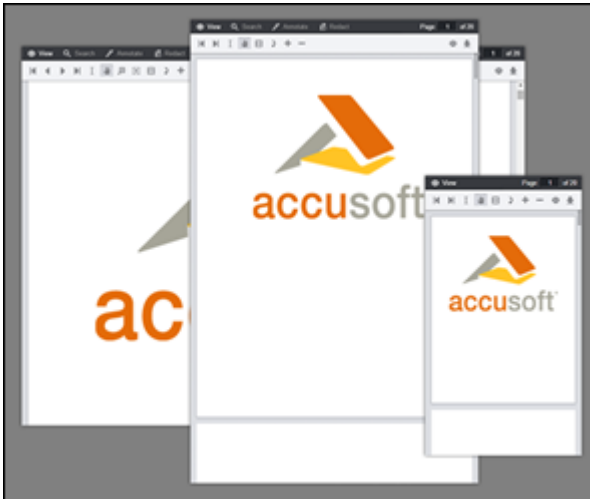
- **Left example above:** embedding the full Viewer using the jQuery plugin.
- **Right example above:** embedding the ViewerControl alone.

The ViewerControl does not have dependencies on third-party libraries.

Design Basics

Introduction

The Viewer interface was designed to adapt to any size viewport. Rather than targeting specific devices, the Viewer will fit to the maximum screen size on any device whether it is a desktop, tablet, or phone.



Media Queries

The Viewer uses CSS3 Media Queries (<http://www.w3.org/TR/css3-mediaqueries/>) with expressions using min-width and max-width to adjust the layout of navigation and dialogs.

Breakpoints

The Media Query Breakpoints, defined in viewer.css, are set according to the Viewer layout. On smaller viewports the tab navigation collapses into a menu and some tools are hidden. On larger viewports the dialogs transform from horizontal to a vertical layout to utilize screen real estate. The breakpoints are as follows:

Example

```
/* Target modern browsers that support media queries */
@media (min-width: 0) {}
/* Mobile & Tablet Sizes, collapse navigation tabs into menu */
@media (max-width: 767px) {}
/* Desktop Sizes */
@media (min-width: 768px) {}
```

Media Queries are not supported in Internet Explorer 8 and no Media Query polyfills are used in this regard. All Internet Explorer 8 specific styles are in legacy.css.

Changing the Breakpoint

To change the breakpoint from the default 768px you will need to change this in two places:

1. In viewer.css, under the comment "viewport breakpoints", look for the following expressions:

Example

```
@media (max-width: 767px) {}  
@media (min-width: 768px) {}
```

2. In viewer.js look for the variable **tabBreakPoint**; this is used in viewer.js to collapse the tab navigation on smaller viewports:

Example

```
this.tabBreakPoint = 767;
```

Legacy Support

Media Queries are not supported in Internet Explorer 8 and no Media Query polyfills are used in this regard. All Internet Explorer 8 specific styles are in legacy.css. Since Media Queries are not supported, if you add styles within a Media Query block in viewer.css you will also need to add this to legacy.css.

Grid System

The Viewer uses a basic grid system to assist with the UI layout. Through a series of rows and columns the layout can scale dynamically. Rows are used to create horizontal groups of columns. Columns are created by defining the number of twelve columns you will span. For example, three columns would use three divs with a class of **.pcc-col-4**:

Example

```
<div class="pcc-row">  
  <div class="pcc-col-4">Left</div>  
  <div class="pcc-col-4">Center</div>  
  <div class="pcc-col-4">Right</div>  
</div>
```

.pcc-col-* classes are active in small viewports and **.pcc-lg-col-*** classes only take effect in larger viewports:

Example

```
<div class="pcc-row">  
  <!--  
  These two divs will span one column on small viewports but  
  split to two columns on larger viewports  
  -->  
  <div class="pcc-col-12 pcc-lg-col-6">Left</div>  
  <div class="pcc-col-12 pcc-lg-col-6">Center</div>  
</div>
```

There are also **.pcc-hide** and **.pcc-show** classes which can be used to toggle content across breakpoints:

Example

```
<div class="pcc-row">
  <!-- This button will only appear on larger viewports -->
  <button class="pcc-hide pcc-show-lg">Left</button>
  <!-- This button will only appear on smaller viewports -->
  <button class="pcc-show pcc-hide-lg">Center</button>
</div>
```

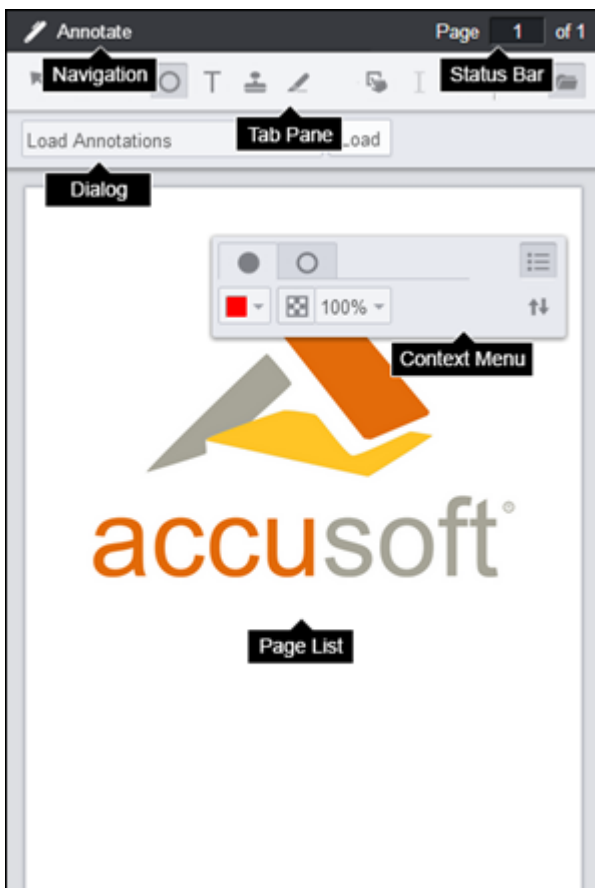
Polyfills

There are a few polyfills used to provide support for browser features:

- **HTML5 Shiv** (<https://github.com/aFarkas/html5shiv>) - The HTML5 Shiv enables use of HTML5 sectioning elements in legacy Internet Explorer and provides basic HTML5 styling for Internet Explorer 6-9, Safari 4.x (and iPhone 3.x), and Firefox 3.x.
- **Normalize** (<http://necolas.github.io/normalize.css/>) - Normalize provides better cross-browser consistency in the default styling of HTML elements.

Components

The Viewer is made up of a number of UI components:



- Tab Navigation - The set of tabs that distinguishes different aspects of the Viewer functionality.
- Tab Pane - The tools specific to each tabset. This can be configured to display horizontally or vertically.
- Status Bar - Displays the page number and allows you to jump to a specific page.
- Dialog - Menu area for extended options and settings.
- Context Menu - Menu that allows you to change properties of annotations.
- Page List - The viewer control that renders the document.

The styles for Tab Navigation, Tab Pane, Status Bar, Dialog, and Context Menu are defined in viewer.css. The styles for Page List are defined in viewercontrol.css.

Templates

You can change the markup of the Viewer UI components by editing the templates. The templates are HTML files ending in *.Template.html. The templates are consumed using the Underscore.js Template utility function. Variables and JavaScript conditions can be used within the templates using ERB syntax. For more information see the Underscore documentation at <http://underscorejs.org/#template>.

For a complete list of templates, refer to the [HTML Templates](#) topic.

Disabling Tabs

To disable one of the navigation tabs you could comment out the HTML in the templates or pass one of the following configuration parameters to the jQuery viewer plugin:

Example

```
var pluginOptions = {
  uiElements: {
    redactTab: false
  }
};
```

Architecture Basics

Architecture Basics

The Viewer has a multi-tier architecture, which is used to achieve a simple out-of-the box and customizable experience.

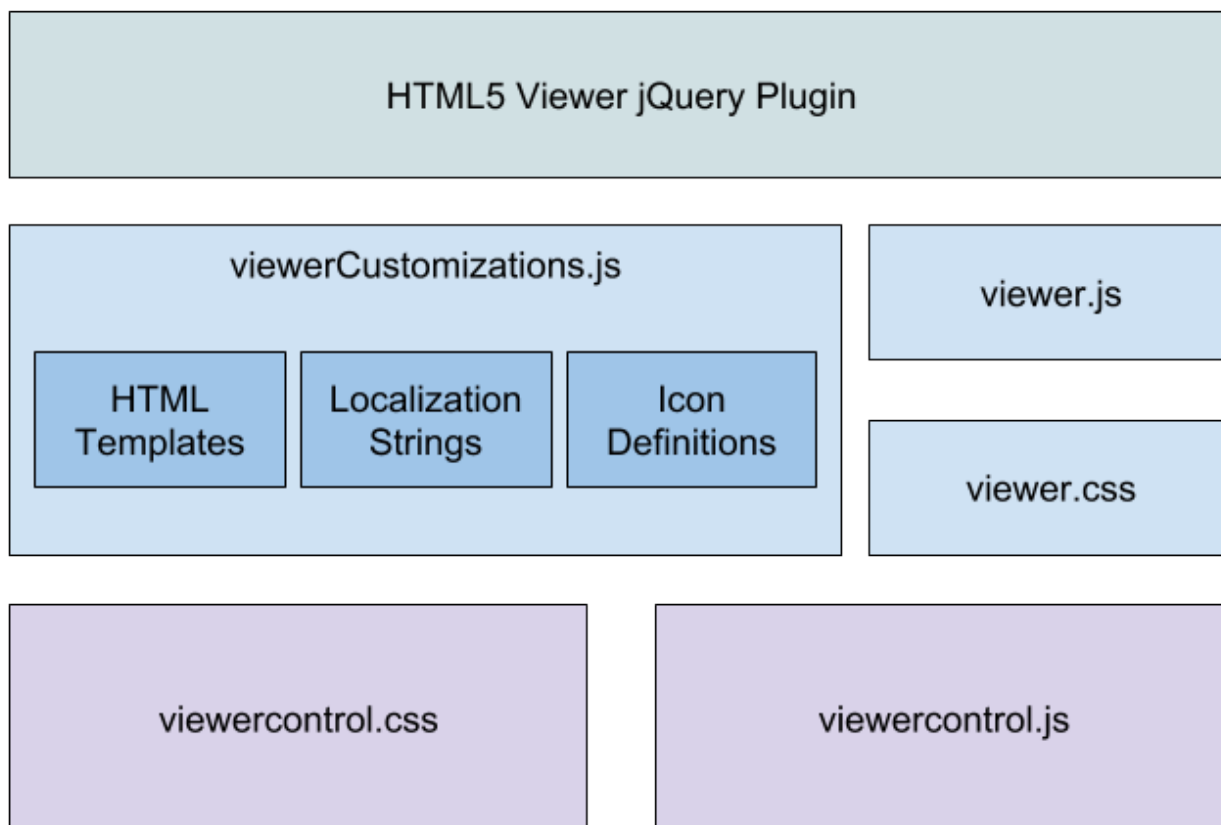
The jQuery Plugin

At a high level, the Viewer is delivered as a configurable jQuery plugin. When using the jQuery plugin, the caller needs a basic understanding of jQuery selectors and the ability to copy and paste from sample code.

An understanding of the Viewer architecture is not required for the out-of-the box experience offered by the jQuery plugin.

Beyond the jQuery Plugin

- The Viewer is built through the jQuery plugin using several open-source CSS, JavaScript, and HTML template files.
- These files implement the Viewer UI-chrome, which includes all of the Viewer tabs, buttons, dialogs, and inputs.
- The open-source Viewer UI-chrome builds on top of the ViewerControl, which displays the document.
- Files that implement the Viewer chrome may be customized in order to customize the Viewer UI.



Separation of Concerns

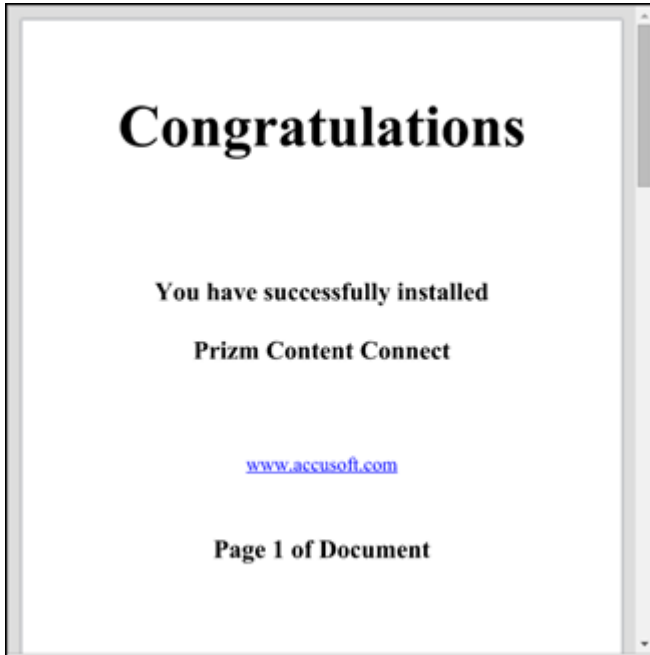
We apply the principle of separation of concerns within the Viewer implementation, in order to promote customization of the code.

Aspects of the UI code use MVC concepts. The code in viewer.js acts as a UI controller (or mediator) between the DOM (view) and the ViewerControl (model). To achieve this pattern, viewer.js leverages third party tools such as jQuery and Underscore.js' templating system.

All markup for the UI-chrome is written in the HTML template files.

ViewerControl

The ViewerControl is a core component to any viewer. It implements the logic of document display, mouse tools and touch interaction, search, printing, annotations, and redactions. It is responsible for calling to the PCCIS services via the web tier, to retrieve document and annotation data. It renders a UI - the page list - which permits scrolling through the content of a document.



ViewerControl is Reusable

The ViewerControl is a reusable component that can be instantiated by itself. This gives a chrome-less viewer, which exposes the full API. Building a custom Viewer UI around the ViewerControl is one approach to building highly custom viewers.

ViewerControl API

The ViewerControl object exposes an API that gives access to the state of the Viewer and can be used for programmatic control over all functions of the ViewerControl. This API is consumed by any code that builds on top of it in order to create rich UI.

ViewerControl is Accessible Through the jQuery Plugin

You don't have to directly embed the ViewerControl to access its API. The ViewerControl object is accessible when embedding the Viewer with the jQuery plugin, and its API can be used to control the Viewer created by the jQuery plugin.

```
var viewerControl = $("#myDiv").pccViewer(pluginOptions).viewerControl;  
  
viewerControl.on("PageCountReady", function(){  
    viewerControl.changeToLastPage();  
});
```

Integrate PrizmDoc Viewer Releases with Your Code

Integrate PrizmDoc Releases with Your Code

The Viewer offers complete customizability. However, performing customization adds additional maintenance costs. If you choose to perform significant customization, consider how you will integrate future releases of the Viewer with your code.

You can choose to run the default Viewer with no changes outside of configuration parameters passed in to the Viewer in your web tier. This allows for the simplest integration with subsequent releases. More likely, you will want to customize the CSS, HTML, and even the unobfuscated JavaScript libraries to ensure that the Viewer meets the needs of your product.

Below are a few guidelines to help you make the right decisions in customizing your Viewer:

Viewer API ([viewerControl.js](#))

The Viewer API is the base building block of the Viewer. We ensure that API changes are backward compatible with point releases (for example, PrizmDoc v11.1 → PrizmDoc v11.2) and will not introduce breaking changes unless critical. With major releases we also endeavor to ensure backward compatibility with previous releases of the Viewer API.

HTML Templates and CSS

The Viewer that is shipped with the product will be maintained and enhanced from release to release. The Viewer HTML and CSS markup will change with each release. Once you have begun to modify your markup, it is recommended that you consider subsequent PrizmDoc releases as sample code, in which you would evaluate product changes and choose to incorporate all or parts of those changes into your customization.

Viewer Resources ([viewerCustomizations.js](#))

The Viewer can be customized at compilation time with custom icons, translations, and the above HTML Templates, all provided at initialization time via [viewerCustomizations.js](#).

JavaScript files ([viewer.js](#))

The Viewer JavaScript that lies above the Viewer API is unobfuscated and open for customization. While we expect many developer needs will be satisfied through configuration parameters and minor HTML or styling changes, some developers will desire to modify [viewer.js](#) for more advanced customization. You should carefully consider your development and ongoing maintenance strategy to ensure that future releases of PrizmDoc are easy to integrate into your customizations. We cannot guarantee backward compatibility of [viewer.js](#) in future releases as it is central to the functionality of the Viewer.

Configure the Viewer

This section contains the following information on how to configure the Viewer. You can also see [code examples](#) on our website that demonstrate how to configure the Viewer.

- [Configuration Options](#)
 - [Initialization Parameters](#)
 - [uiElements](#)
 - [Use Pre-Loaded Search Parameters](#)
 - [Configure the Skinny Comments Panel](#)
 - [Define the View Mode](#)
 - [Digital Rights Management Configuration](#)
 - [Enable Content Encryption](#)
 - [How to use Pre-Defined Search](#)

- [Localize the Viewer](#)
- [Use a Custom Resource Path](#)
- [Add Custom Image Stamps](#)

Configuration Options

This section contains the following information:

- [Initialization Parameters](#)
- [uiElements](#)
- [Use Pre-Loaded Search Parameters](#)
- [Configure the Skinny Comments Panel](#)
- [Define the View Mode](#)
- [Digital Rights Management Configuration](#)
- [Enable Content Encryption](#)
- [How to use Pre-Defined Search](#)
- [Localize the Viewer](#)
- [Use a Custom Resource Path](#)
- [Add Custom Image Stamps](#)

Initialization Parameters

For a complete list of configuration options when initializing the Viewer plugin, refer to the [Namespace: fn](#) API topic.

uiElements

For a complete list of configurable UI elements when initializing the Viewer plugin, refer to the [Namespace: fn](#) API topic.

Use Pre-loaded Search Parameters

preDefinedSearch Parameters

This object contains all predefined search options:

Parameter	Data Type	Description
<code>highlightColor</code>	String	The default highlight color of the search terms. This is overridden by the <code>term-level</code> parameter. This must be in 6 digit hexadecimal format preceded by a <code>#</code> . Example: <code>"#ee3a8c"</code>
<code>searchOnInit</code>	Boolean	Run search on launch.
<code>globalOptions</code>	Object	Set the default search options for each of the predefined search terms. This is overridden by the term-level <code>options</code> parameter. Example:

Parameter	Data Type	Description
		<pre>predefinedSearch : { globalOptions: { matchCase: false, endsWith: false, beginsWith: false, matchWholeWord: false } }</pre>
<code>terms</code>	Array	<p>An array of objects that represent the search terms that will be available in the predefined search menu.</p> <p>Example:</p> <pre>predefinedSearch : { terms: [{ searchTerm: "llama" }] }</pre>

predefinedSearch Terms

This object represents the search terms that will be available in the predefined search menu:

Parameter	Data Type	Description
<code>searchTerm</code>	String	The search string for the term object. This is overridden by the <code>userDefinedRegex</code> parameter.
<code>userDefinedRegex</code>	String	<p>A regular expression that will be searched in place of <code>searchTerm</code>. The first and last forward slashes, as well as the flags, are stripped from the string. For example, <code>"/Pa(\\w+)/ig"</code> will become <code>"Pa(\\w+)"</code>. When special characters (ex: backslash) are used in the <code>userDefinedRegex</code> field, they need to be properly escaped. For example, for searching words that begins with "Pa", the regular expression will be <code>"Pa(\\w+)"</code>, this regular expression should be properly escaped like this <code>"Pa(\\w+)"</code>. All patterns use the Global(g) flag.</p> <p>Example:</p> <pre>predefinedSearch : { terms: [{ searchTerm: "4 digits" </pre>

Parameter	Data Type	Description
		<pre> userdefinedRegex: "(\\d{4})" }] } </pre>
<code>selected</code>	Boolean	Whether or not this term will be selected in the menu.
<code>options</code>	Object	<p>Set the search options for this term. If a parameter is not defined it will inherit the <code>globalOptions-level</code> parameter.</p> <p>Example:</p> <pre> predefinedSearch : { terms: [{ searchTerm: "Lla," options: { matchCase: true, endsWith: false, beginsWith: true, matchWholeWord: false } }] } </pre>

Configure the Comments Panel

Introduction

To better accommodate the Comments view for both small and large viewers, the Viewer automatically switches between displaying the full comments and displaying only a Comments icon. When you click the Comments icon, the full comment is expanded.

The `commentsPanelMode` options are:

- **full** - The entire content of the comments are displayed in the sidebar of the document.
- **skinny** - An icon is placed in the sidebar of the document, representing each comment thread. When the icon is clicked, the comment thread is expanded to show the full content.
- **auto** - This mode will intelligently switch between the full and skinny mode, in order to optimize the space available for viewing the document. The default is "auto" when the option is not specified.

The examples below show the use of the `commentsPanelMode` configuration parameter.

Forcing the Viewer to always display skinny comments:

Example

```
var pluginOptions = {
  documentID: viewingSessionId,
  language: languageItems,
  template: htmlTemplates,
  commentsPanelMode: "skinny"
};
$(document).ready(function () {
  var viewerControl = $("#viewer1").pccViewer(pluginOptions).viewerControl;
});
```

Forcing the Viewer to always display full comments:

Example

```
var pluginOptions = {
  documentID: viewingSessionId,
  language: languageItems,
  template: htmlTemplates,
  commentsPanelMode: "full"
};
$(document).ready(function () {
  var viewerControl = $("#viewer1").pccViewer(pluginOptions).viewerControl;
});
```

Allowing the Viewer to choose between skinny and full comments, which is the default option:

Example

```
var pluginOptions = {
  documentID: viewingSessionId,
  language: languageItems,
  template: htmlTemplates,
  commentsPanelMode: "auto"
};
$(document).ready(function () {
  var viewerControl = $("#viewer1").pccViewer(pluginOptions).viewerControl;
});
```

Define the View Mode

Introduction

The `ViewMode` enumeration defines view modes known by `PCCViewer.ViewerControl`. The `ViewerControl` uses a specified view mode to set or update how documents that contain different sized pages are displayed in the Viewer. This enumeration is a convenience for API developers. Instead of using it, you can pass in the string values of the view mode (enumeration values) directly to the API.

There are three view modes available:

- **Document** - The Viewer maintains the relative size of each page when displaying a document. For example, if page 2 is smaller than page 1, it will appear smaller. In this view mode, the user can choose to scroll through the document or select one of the View icons to go from the First page to the Last page of the document.
- **EqualWidthPages** - The Viewer scales each page so that their width is the same. For example, if page 2 is smaller than page 1, it will be scaled larger so that its width is equal to the width of page 1. In this view mode, the user can choose to scroll through the document or select one of the View icons to go from the First page to the Last page of the document.
- **SinglePage** - The Viewer displays a single page at a time. Each page is scaled to fit within a view box, which is the initial size of the Viewer and increases in size when zooming in (and decreases in size when zooming out). After the Viewer initializes, the view mode may not be changed to or from `SinglePage` view mode (or an exception will occur). In this view mode, the user selects one of the View icons to go from the First page to the Last page of the document which simulates a smooth transition from page to page. This view mode is non-scrolling.

To set the `ViewMode`:

Example

```
// use the enumeration
myViewerControl.setViewMode(PCCViewer.ViewMode.SinglePage);
// or just use the string value
myViewerControl.setViewMode("SinglePage");
```

Digital Rights Management Configuration

Introduction

The Viewer can be configured to disable UI buttons that will allow an end user to easily duplicate the content of a document.

The following UI buttons can be disabled using configuration options:

- **Download [document] button** - Hide the button to download the original document.
- **Select text button** - Hide the button to select the text selection mouse tool, which inhibits the user's ability to select and copy selected text.
- **Print button** - Hide the button to print the document.

Example

```
// DRM options are controlled through the viewer's options argument. var
pluginOptions = {
  documentID: viewingSessionId,
  language: languageItems,
  template: htmlTemplates,
  uiElements: {
    download: false, // hide download button
    copyPaste: false, // hide select text tool button
    printing: false // hide print button
  },
};

$("#myDiv").pccViewer(pluginOptions);
```

Server-Side DRM

DRM options for the Viewer are enforced only in the Viewer UI. A skilled end user can manipulate the browser to circumvent the viewer-based DRM enforcement.

Techniques a skilled user can use to circumvent viewer-based DRM enforcement:

1. Edit the JavaScript run by the browser, which allows them to:
 - Change the plugin options for DRM.
 - Directly call the API of the viewer control to print or set the select text tool.
2. Directly call the server API to download the original document.

Additional security measures can be added using server-side code changes which are listed below:

Document Download

1. Create a new **viewerTemplate.html** file that excludes the document download button (**data-pcc-download**).
 - Using this technique, the download button will not be available, regardless of the plugin options.

Copying Text

There are not any server-side techniques to strengthen DRM enforcement of copying text. However, removing the text selection control from the UI will require the user to understand the text selection API in order to enable it on the Viewer. The manner in which the product renders SVG also makes it nearly impossible to copy text just using a browser's text selection capability.

Printing

1. Create a new **viewerTemplate.html** file that excludes the print button (**data-pcc-print="launch"**).
 - Using this technique, the print button will not be available, regardless of the plugin options.
2. Exclude the print template from the configuration object passed to the Viewer (**pluginOptions.template.print**).
 - This can be controlled by the server-side code that generates the page.
 - Using this technique, the **ViewerControl#print(options)** method will be non-functional.

Content Encryption

For an added layer of security, Content Encryption can be enabled to provide an obscured transfer of data from the PrizmDoc Server to the Viewer website, preventing unauthorized agents from discerning the content being transmitted. See [Enabling Content Encryption](#) for more information.

PrizmDoc Viewer is not designed or intended to be a fail-proof DRM system but does provide a few basic security measures to prevent most users from unintentionally accessing content to which they are not authorized.

Enable Content Encryption

Introduction

This topic contains an overview and steps to help you enable content encryption:

- [Overview of Enabling Content Encryption](#)
- [Enabling Content Encryption in PrizmDoc Server via the Central Configuration File](#)

- [Enabling Content Encryption in PrizmDoc Server via ViewingSession Property](#)
- [Enabling Content Encryption in the Viewer](#)

The goal of content encryption is to provide an obscured transfer of data from the PrizmDoc Server to the Viewer website, preventing unauthorized agents from discerning the content being transmitted. Additional security can be enabled by configuring the Viewer and server to communicate over the Secured Socket Layer (SSL), https protocol, rather than standard non-secure http protocol. In cases where this is not viable or enough protection, the content encryption adds a strong measure of privacy to the document content. When content encryption is enabled, the web data images and document text strings sent to the Viewer will be encrypted and then decrypted by the Viewer.

Overview of Enabling Content Encryption

Content encryption must be enabled in the Viewer and in the PrizmDoc Server; it is disabled by default. Enabling content encryption in the Viewer is straightforward and performed by an option passed to the Viewer constructor or jQuery plugin. This process is documented below.

The file paths for the Central Configuration file are:

- **Linux:** /usr/share/prizm/prizm-services-config.yml
- **Windows:** C:\Prizm\prizm-services-config.yml

NOTE: The default installation directory is: C:\Prizm.

There are two options for enabling content encryption on the server:

1. **Enable content encryption via the central configuration file (prizm-services-config.yml** - located in the top-level of the installation directory): this enables content encryption for all viewing sessions.
2. **Toggle (enable or disable) content encryption via viewing session property:** this enables or disables content encryption per viewing session, overriding the option set in the central configuration file.

These two options are documented below.

NOTE: For security reasons, toggling content encryption per viewing session is not permitted in the out-of-box product configuration. It must be explicitly allowed via the [central configuration file](#).

Finally, it's important to note it must be enabled or disabled on both the Viewer and server, or unexpected behavior will occur. If encryption is enabled on the server but not for the Viewer, then the content will not be rendered correctly. If encryption is enabled for the Viewer but not on the server, then the content will not be encrypted during transit, however, it will be rendered correctly in the Viewer.

In summary:

- Content encryption is disabled out of the box.
- It must be enabled in the Viewer and PrizmDoc Server.
- It can be enabled or disabled on the server via the central configuration file.
- If permitted, enabling or disabling content encryption can be overridden when creating a viewing session.

Enabling Content Encryption in PrizmDoc Server via the Central Configuration File

To enable content encryption follow the steps below:

1. Open the central configuration file, **prizm-services-config.yml** in your favorite editor. The prizm-services-config.yml file is located in the top-level of the installation directory.

2. Find the **viewing.contentEncryption.enabled** section and change the value to **true**.

Encrypted Transmission Example

```
# Controls whether or not content is encrypted by the back end before being
# transmitted to a client viewer. The client viewer will decrypt the content
# in
# the browser. This is useful for DRM, making it more difficult to copy
# protected content that has been delivered to the browser.
#
viewing.contentEncryption.enabled: true
```

3. Save the **changes** to the file.
4. Restart the **PrizmDoc Server** for the changes to take effect.
5. Continue by enabling the **encryption option** for the Viewer as described in the section below.

Enabling Content Encryption in PrizmDoc Server via the ViewingSession Property

1. Open the central configuration file, **prizm-services-config.yml** in your favorite editor. The **prizm-services-config.yml** file is located in the top-level of the installation directory.
2. Find the **viewing.sessionConstraints.pageContentEncryption.allowedValues** section and change the value to ["**default**", "**enabled**", "**disabled**"].

Encrypted Transmission Example

```
# Defines the list of allowed values for the pageContentEncryption viewing
# session creation option.
#
# Must be an array with either ONE or ALL of the following strings:
#
# "default" - Allow REST API callers to create a new viewing session
without
# explicitly stating whether or not page content encryption
(DRM)
# should be applied. The value configured in this file at
# viewing.contentEncryption.enabled will be used to determine
# whether or not page encryption is applied.
#
# "enabled" - Allows REST API callers to explicitly enable page content
encryption (DRM) when creating a new viewing session,
overriding
# whatever value is configured in this file by
# viewing.contentEncryption.enabled.
#
# "disabled" - Allows REST API callers to explicitly disable page content
encryption (DRM) when creating a new viewing session,
overriding
# whatever value is configured in this file by
# viewing.contentEncryption.enabled.
#
viewing.sessionConstraints.pageContentEncryption.allowedValues:
["default", "enabled", "disabled"]
```

3. Save the **changes** to the file.
4. Restart the **PrizmDoc Server** for the changes to take effect.
5. Update your **web-tier code** to set the value of the **pageContentEncryption** Viewing Session property to **"enabled"** when creating the viewing session. The example below is for a .NET web tier:

Example

```
viewingSessionProperties.pageContentEncryption = "enabled";
....
// Serialize document properties as JSON which will go into the body of the
request string requestBody = serializer.Serialize(viewingSessionProperties);
requestStream.Write(requestBody);
```

6. Continue by enabling the encryption option for the Viewer as described in the section below.

Enabling Content Encryption in the Viewer

To enable encryption in the Viewer, provide the encryption option in the viewer options parameter as follows so that the Viewer can handle encrypted data:

Example

```
<script type="text/javascript">
$(function() {
    $('#viewerContainer').pccViewer({
        documentID: 'XYZ...',
        imageHandlerUrl: '/pas-proxy',
        viewerAssetsPath: 'viewer-assets',
        resourcePath: 'viewer-assets/img',
        language: viewerCustomizations.languages['en-US'],
        template: viewerCustomizations.template,
        icons: viewerCustomizations.icons,
        annotationsMode: "LayeredAnnotations",
        encryption: true
    });
});
</script>
```

Enabling the encryption will not work without setting the configuration parameter as described above. Also, if the PrizmDoc Server configuration setting is either not set or the PrizmDoc Server is not restarted, the data will arrive unencrypted.

How to Start & Stop the PrizmDoc Server

Refer to these topics for additional information:

- **Windows:** [Installation Guide > Starting & Stopping the PrizmDoc Server > Windows](#)
- **Linux:** [Installation Guide > Starting & Stopping the PrizmDoc Server > Linux](#)

How to Use Pre-defined Search

Introduction

This feature allows you to define a set of predefined search terms. To enable this functionality you must add the **predefinedSearch** property to the Viewer parameters. The following example shows you how:

Example

```
<script type="text/javascript">
  $(document).ready(function () {
    var pluginOptions = {
      documentID: viewingSessionId,
      language: languageItems,
      template: htmlTemplates,
      predefinedSearch: {
        highlightColor: "#ee3a8c",
        searchOnInit: false,
        globalOptions: {
          matchCase: false,
          endsWith: false,
          beginsWith: false,
          matchWholeWord: false,
          wildcard: false
        },
        terms: [{
          searchTerm: "llama",
          selected: true,
          options: {
            matchWholeWord: true,
            wildcard: false
          }
        },
        {
          searchTerm: "Words that begin with ll",
          userDefinedRegex: "\\bll(\\w*)\\b",
          searchTermIsRegex: true,
          selected: true,
          highlightColor: "#4169e1",
          options: {
            matchCase: true
          }
        }
      ]
    }
  });
  $("#sample").pccViewer(pluginOptions);
}</script>
```

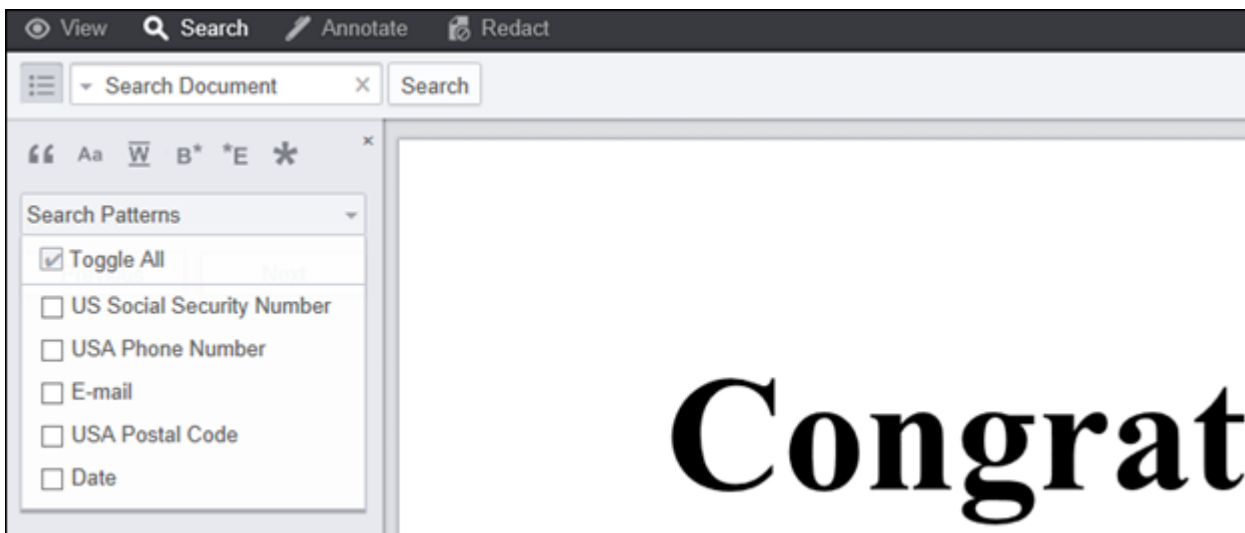
PredefinedSearch.JSON

Predefined Search can also be specified using a text file (`predefinedSearch.json`). The `predefinedSearch.json` file provides several sample search terms and custom regular expressions; the file is

parsed by the web-tier and loaded in the Viewer. The following example shows you how:

Example

```
<script type="text/javascript">
var viewingSessionId =
'<%=HttpUtility.JavaScriptStringEncode(viewingSessionId)%>';
//Retrieve the searchJson (search data) into javascript var searchTerms =
<%=searchJson%>;
var pluginOptions = {
    documentID: viewingSessionId,
    predefinedSearch: searchTerms,
};
$(document).ready(function () {
    var viewerControl = $("#viewer1").pccViewer(pluginOptions).viewerControl;
});
</script>
```



Predefined Search Patterns

Parameter	Data Type	Description
<code>highlightColor</code>	String	The default highlight color of the search terms. This is overridden by the term-level parameter. This must be in 6 digit hexadecimal format preceded by a #. Example: "#ee3a8c"
<code>searchOnInit</code>	Boolean	Run search on launch.
<code>globalOptions</code>	Object	Set the default search options for each of the predefined search terms. This is overridden by the term-level <code>options</code> parameter. Example: <pre>predefinedSearch : { globalOptions: {</pre>

Parameter	Data Type	Description
		<pre> matchCase: false, endsWith: false, beginsWith: false, matchWholeWord: false, wildcard: false } } </pre>
<code>terms</code>	Array	<p>An array of objects that represent the search terms that will be available in the predefined menu.</p> <p>Example:</p> <pre> predefinedSearch : { terms: [{ searchTerm: "llama" }] } </pre>

Predefined Search Terms

Parameter	Data Type	Description
<code>searchTerm</code>	String	The search string for the term object. This is overridden by the <code>userDefinedRegex</code> parameter.
<code>searchTermIsRegex</code>	Boolean	When set to true will use <code>userDefinedRegex</code> to execute the search.
<code>userDefinedRegex</code>	String	<p>A regular expression that will be searched in place of <code>searchTerm</code>. The first and last forward slashes, as well as the flags, are stripped from the string. For example, <code>"/Pa(\w+)/ig"</code> will become <code>"Pa(\w+)"</code>.</p> <p>When special characters (for example, backslash) are used in the <code>userDefinedRegex</code> field, they need to be properly escaped. For example, for searching words that begin with "Pa", the regular expression will be <code>"Pa(\w+)"</code>. This regular expression should be properly escaped like this: <code>"Pa(\w+)"</code>.</p> <p>All patterns use the Global(g) flag.</p> <p>Example:</p> <pre> predefinedSearch : { terms: [</pre>

Parameter	Data Type	Description
		<pre>{ searchTerm: "4 digits" userdefinedRegex: "(\\d{4})" }]</pre>
<code>description</code>	String	Description of the search term. If description is not defined, then <code>searchTerm</code> will be used.
<code>highlightColor</code>	String	When specified the system will use this value to show the highlight color for this search term. When not specified the system will generate a color. Example: <code>highlightColor: "#FFFF20"</code>
<code>Options</code>	Object	Example: <pre>options: { "matchCase": false, "endsWith": false, "beginsWith": false, "matchWholeWord": false, "exactPhrase": false, "wildcard": false }</pre>

Localize the Viewer

Introduction

The Viewer has a language localization feature which allows you to customize labels and text using a language JavaScript object. The language object is passed as a parameter in the Viewer plugin configuration options:

Example

```
var pluginOptions = {
  documentID: viewingSessionId,
  language: languageItems,
  template: htmlTemplates
}
```

Using the Language Parameters

In viewer.js the language parameters are used in various places. The following example shows a message with the

`printRangeError` language parameter:

Example

```
viewer.notify({message: viewer.language.printRangeError});
```

When the HTML templates are loaded, using the Underscore.js Template utility function, the language object is used as template data:

Example

```
element.html(_.template(options.template.viewer, options.language))
```

The language parameters are then referenced as variables in the templates:

Example

```
<!-- This is using the "rotate" parameter from language.json -->
<button data-pcc-rotate class="pcc-icon pcc-icon-rotate"
  title="<%= rotate %>"></button>
```

For more information on template syntax see the Underscore.js Template documentation at <http://underscorejs.org/#template>.

Common Pitfalls

When editing the `language.json` file or the templates there may be some errors for which the cause may not be immediately obvious. Here are some common console errors and their possible causes:

Uncaught TypeError

- **Cannot read property `languageElements` of undefined** - This error is thrown when `viewer.js` cannot find the language object. Verify that the `language.json` file is being read and parsed correctly.

Uncaught ReferenceError

- **x is not defined** - This error could be thrown if you have referenced a variable in the HTML templates that is not defined as data when loading the template. Check to see that this variable exists in either the `language.json` file or as a template data property used when loading a template.

Use a Custom Resource Path

Introduction

In some instances you may want to change the path that the `viewercontrol.js` uses to look for images. For example, instead of **img** you would like to use **images**. To achieve this, add the `resourcePath` parameter to the

Viewer plugin configuration options and specify the new path.

The trailing slash is not required and this path can be absolute or relative. Some examples of valid paths are:

- "img"
- "../img"
- "/PrizmCC_HTML5_Viewer_CS/html5/img"
- `http://prizmdemos.accusoft.com/PCCv9Preview2/html5/img`

Example

```
var pluginOptions = {
  documentID: viewingSessionId,
  language: languageItems,
  template: htmlTemplates,
  resourcePath: 'images'
}
```

If you change the image directory name you will also need to update the background image URIs in the `Viewer.css`.

The `resourcePath` is used in the obfuscated code to obtain images like the `markhandles`. You can work around the hard coded paths in the `Viewer.css` by editing the `Viewer.css`.

For example, you can change the folder to another location on your file system within the same domain. If you want to put the images in a folder called "imgServer1", you can provide `resourcePath : "imgServer1"` and edit the `Viewer.css` as shown in the following example:

Example

```
.pccv .pccEditMarkButton {
  background-image: url(../imgServer1/EditTextMark@2x.png);
}
```

Add Custom Image Stamps

Introduction

You can customize the image stamps available to Viewer users by adding or deleting image files from the ImageStamp folder, located by default in:

- **Windows** - C:\ProgramData\Accusoft\Prizm\ImageStamp
- **Linux** - /usr/share/prizm/Samples/imageStamp

By default, two image stamps are included in the installation: a green 'checkmark' and a red 'x'. You can choose to leave these in place, or delete them and add image files of your choosing. The Viewer supports the following file extensions for image stamps by default:

- PNG
- JPG
- JPEG

- GIF

Example

```
imageStamps.validTypes: ["png", "jpg", "jpeg", "gif", "svg", "webp", "tiff", "tif"]
```

To add more supported file types, refer to the [PrizmDoc Application Services \(PAS\)](#) configuration options.

How to Customize the Viewer

Introduction

The Viewer can be customized in a variety of ways, from quick integrations with minimal configuration to complete control over the Viewer API. Tabs and basic DRM functions such as printing, text selection, and document download can be quickly hidden using configuration parameters.

It's very simple to reorganize menus, add/remove tabs, and customize the look of the Viewer by editing the markup and .css. All Viewer functionality is built on top of the Viewer API, allowing complete control over all Viewer functions.

Configuration Options

The Viewer UI and behavior can be configured when the Viewer is embedded by using JavaScript parameters. For example, you can set the following:

- Displaying/Hiding Tabs and DRM functions
- Enabling Content Encryption
- Using a Custom Resource Path
- Localizing the Viewer
- How to use Predefined Search
- Digital Rights Management Configuration

Customizing Markup and Styles

If the Viewer needs to be customized more than the configuration options allow, all UI code is open-source and can be modified to change the following:

- Reorganizing Menus
- Creating a Custom Tab
- Changing annotation default values
- Changing the position of the menu bar
- Customizing Styles (css)

Viewer API

The Viewer API permits programmatic control over the Viewer. The API allows callers to augment, customize, or automate the end user's experience with the Viewer. Functionality that is exposed through the Viewer API includes:

- Creating and destroying the Viewer
- Events

- Page navigation
- Zooming and fitting content
- Mark (annotation and redaction) CRUD
- Markup saving and loading
- Customizing mouse tools
- Searching document text
- Printing
- Getting page and document attributes

In addition to the code samples provided in this section, we have a number of interactive, live code examples on our [website](#).

Add a Custom Button

Introduction

Adding a custom button is as simple as adding some markup and a little bit of JavaScript anywhere on the web page. In this example below, we will look at adding a button to the Viewer in a way that matches the design language and code style of the existing Viewer.

If you are not yet familiar with the code structure, refer to these topics:

- [Modifying viewer.js](#)
- [Creating a Custom Tab](#)

In this topic, we will add a button to the default **View tab** which will add an **Approved stamp** annotation to the top right of the first page in the document.

Adding the Button HTML

The bulk of the Viewer markup is inside the file **viewerTemplate.html**; this includes all the toolbars and vertical slide-outs.

***NOTE:** If you would like to add buttons to the context menu, the annotations saving dialog, or the print dialog, these are found in separate files.*

The **View tab** appears at the top of the document, and is identified by the data attribute **data-pcc-nav-tab="view"**. Inside the **.pcc-tab-pane**, the actual menu bar, there are two lists of buttons. One is the normal list starting from the left, **.pcc-left**, and the second is the buttons floating on the right side, **.pcc-pull-right**. To add the button on the right side:

Example

```
<div class="pcc-pull-right">
  <button class="myCustomApprovedButton">Approve</button>
  <button class="pcc-icon pcc-icon-print" data-pcc-print="launch"></button>
  <button data-pcc-download class="pcc-icon pcc-icon-download"></button>
</div>
```

Note that the actual list of other buttons in the **.pcc-pull-right** section may be different from the example. However, adding the example code for the button with the class **"myCustomApprovedButton"** to the top of the list will make it appear first in the right-hand side buttons.

Adding the Custom JavaScript

Associating logic to the buttons is handled in the **viewer.js** file.

First, find the button in the DOM. Toward the top of the file, there is a property on the Viewer, **this.viewerNodes**, which holds all of the Viewer DOM elements. Add the button to the end of the list:

Example

```
...
$searchBeginsWith: viewer.$dom.find("[data-pcc-search=beginsWith]"),
$searchEndsWith: viewer.$dom.find("[data-pcc-search=endsWith]"),
$searchWildcard: viewer.$dom.find("[data-pcc-search=wildcard]"),
$customApproved: viewer.$dom.find(".myCustomApprovedButton")
};
```

We have used a variable name starting with a **\$**, the global name of jQuery, to indicate that this is a jQuery-wrapped variable. Any time a variable name starts with a **\$** in this file, it indicates that the object is able to use the entire jQuery API.

Second, add logic to the button's click event. A few lines down from this section is the function **bindMarkup**, where logic is added to the DOM nodes. To keep with convention, go all the way to the bottom of this function, and add the click handler there:

Example

```
viewer.viewerNodes.$customApproved.on("click", function (ev) {
  // get the first page attributes
  viewer.viewerControl.requestPageAttributes(1).then(
    function success(attributes) {
      // let's add a stamp now
      var mark = viewer.viewerControl.addMark(1, "StampRedaction");
      // set the stamp text
      mark.setLabel("Approved");
      // set the stamp location
      mark.setRectangle({
        width: 180,
        height: 50,
        x: attributes.width - 200,
        y: 20
      });
    },
    function failed(error) {
      // :( tell the user there was an error
      alert(error);
    }
  );
});
```

Styling the Button

The default Viewer SVG icons are in the `/viewer-assets/src/icons/svg` folder of each sample. For this example, let's assume that you have an icon already created, named **custom-check.svg**.

You will need to add your SVG file to the `/viewer-assets/src/icons/svg` folder and specify **id="pcc-icon-custom-check"** and **viewBox="0 0 52 52"** in the SVG. Note that the id must begin with "pcc-icon-". Copy your SVG path content to the SVG symbol element you added.

Then simply update your viewer HTML template to use this new icon by adding the **pcc-icon** and **pcc-icon-custom-check** class to your button:

Example

```
<button class="myCustomApprovedButton pcc-icon pcc-icon-custom-check"></button>
```

You will then need to build the Viewer assets by running "npm install" and "gulp build". Copy the output `/dist/viewer-assets/js/viewerCustomizations.js` file over the `/viewer-assets/js/viewerCustomizations.js` file in the sample. For more information on building the Viewer assets, see the README file in the viewer-assets folder.

You have completed adding a custom button.

Add Keyboard Shortcuts

Introduction

The Viewer includes support for some keyboard shortcuts. This topic will walk through how the `jQuery.hotkeys` plugin (<https://github.com/jeresig/jquery.hotkeys>) is used to support adding keyboard shortcuts, as well as how easy it is to remove the built-in keyboard support.

The current implementation represents some (but not all) that can be accomplished with keyboard shortcuts. The currently supported keyboard combinations are detailed in the tables below:

Keyboard Key Combinations for Page Navigation

Number	Keyboard Action	Key Combinations	Result
1.	'keydown'	'pageup'	Scrolls the document one page up.
2.	'keydown'	'pagedown'	Scrolls the document one page down.
3.	'keydown'	'home'	document scrolls to the first page
4.	'keydown'	'end'	document scrolls to the last page
5.	"keydown"	'ctrl+g'	puts the cursor in The Viewer's 'go to page' edit box. It allows user to enter the page number to go to.
6.	"keydown"	down arrow	Scrolls the page down.
7.	'keydown'	up arrow	Scrolls the current page up.
8.	'keydown'	left arrow	Scrolls the displayed current page left.
9.	'keydown'	right arrow	Scrolls the displayed current page right.

Zoom in / Zoom out

Number	Keyboard Key Action Type	Key Combinations	Result
1.	'keydown'	'='	zoomin
2.	'keydown'	'-'	zoomout

Delete Selected Marks

Number	Keyboard Key Action Type	Key Combinations	Result
1.	'keydown'	'delete'	Deletes selected marks.

Modal Dialogs

All the following modal dialogs respond to the 'esc' key as if the 'cancel' button was pressed:

1. e-signature dialog
2. Image stamp selection dialog
3. Page redaction dialog
4. Download document dialog
5. Print dialog
6. About box

Number	Keyboard Key Action Type	Key Combinations	Result
1.	'keydown'	'esc'	Closes the dialog. The result is equivalent to pressing the 'cancel' button.

The viewer.js contains a method, **initKeyBindings**, that contains the code to handle the keyboard support as described in the above tables. This method is called in the **initializeViewer** method. In order for the keyboard shortcuts to work, jQuery.hotkeys.min.js file is required.

Example

```
<script src="viewer-assets/js/jquery.hotkeys.js"></script>
```

Do not obtain the file from CDN because it is broken. The non-minified version can be obtained from GitHub: <https://github.com/jeresig/jquery.hotkeys> This is a small file and it is recommended that you read all the details about the plugin before using it in your Viewer.

If you either have your own implementation of the keyboard support or prefer not use this implementation in the viewer.js, simply comment out the call to **initKeyBindings()** in the **initializeViewer** method. Also, you can choose to remove the **initKeyBindings** method definition completely from your copy of the viewer.js.

The following example shows a snippet of code in the method **initKeyBindings** for the 'pageup' key support for scrolling one page up.

Example

```
$('#body').on('keydown', null, 'pageup', function () {
    if ($('#viewer.viewerNodes.$pageList\{0\}`).is(':visible')) {
```



```
        //make sure modals are not up
        if (!$viewer.viewerNodes.$overlayFade\[0\]).is(':visible')) {
            //change to the previous page
            viewer.viewerControl.changeToPrevPage();
            return false;
        }
    }
    return true;
});
```

The following example shows how you can change the code above to trigger the event on the whole document object.

Example

```
$(document).on('keydown', null, 'pageup', function () {
    if ($viewer.viewerNodes.$pageList\[0\]).is(':visible')) {
        //make sure modals are not up
        if (!$viewer.viewerNodes.$overlayFade\[0\]).is(':visible')) {
            //change to the previous page
            viewer.viewerControl.changeToPrevPage();
            return false;
        }
    }
    return true;
});
```

In the above example, line 1 binds the `keydown` action of the `'pageup'` key to the in-line handler. For the Viewer, the node with the selector attribute `'pageList'` is the parent node. Therefore, the handler code checks to see if this node is visible. Also, since you do not want the page navigation to occur when the modal dialogs are showing, in line 4, check for the visibility of the modal dialogs.

Because the elements are `divs` and are not normally focusable, use a wider net and use `'body'` as the target node of the key events. When nothing in particular has focus, `document.body` acts as a target node of key events. You can choose to bind to other elements beside the `'body'` but you may need to give it a tab index. It may not provide expected results in all the browsers. Most browsers have native keyboard focusable support for the following element types:

1. Input elements
2. Buttons
3. link elements

There are other things to consider too. Most browsers provide the following keyboard event types:

1. `'keydown'`
2. `'keyup'`
3. `'keypress'`

The implementation in the Viewer uses `'keydown'` key action. In some cases you may want to use the `'keyup'` event. The `'keypress'` event is not used at all since it is mainly used for capturing key characters entered in the input elements. Not all browsers are consistent in providing key events for all the keys or key combinations. Some browsers will not allow to override their default behavior for a particular key combinations. You may need to experiment/research before choosing key action and key combinations.

The method ***initKeyBindings*** also contains some commented out code that demonstrates how to provide

keyboard support for the buttons in the modal dialogs.

Adding Keyboard Support without using jQuery.hotkeys Plugin

First, set up the Viewer as you would normally.

Example

```
function initKeyBindings (viewerControl) {
  var handler = function(ev){
    return handleGlobalKeypress(ev, viewerControl);
  };

  $(document).on("keydown", handler);
}

var pluginOptions = {
  documentID: viewingSessionId,
  language: languageItems,
  template: htmlTemplates
};

$(document).ready(function () {
  var viewerControl = $("#viewer1").pccViewer(pluginOptions).viewerControl;

  initKeyBindings (viewerControl);
});
```

Example Code for using pageup and pagedown Keys

This code does not check for modal dialogs but the check can be added as shown in the examples above.

Example

```
function handleGlobalKeypress (ev, viewerControl) {
  //check for keys
  switch (ev.keyCode) {
    case 33: // Page Up
      ev.preventDefault();
      viewerControl.changeToPrevPage();
      return false;
    case 34: //Page Down
      ev.preventDefault();
      viewerControl.changeToNextPage();
      return false;
  }
}
```

Build a Custom User Interface

Build a Custom User Interface

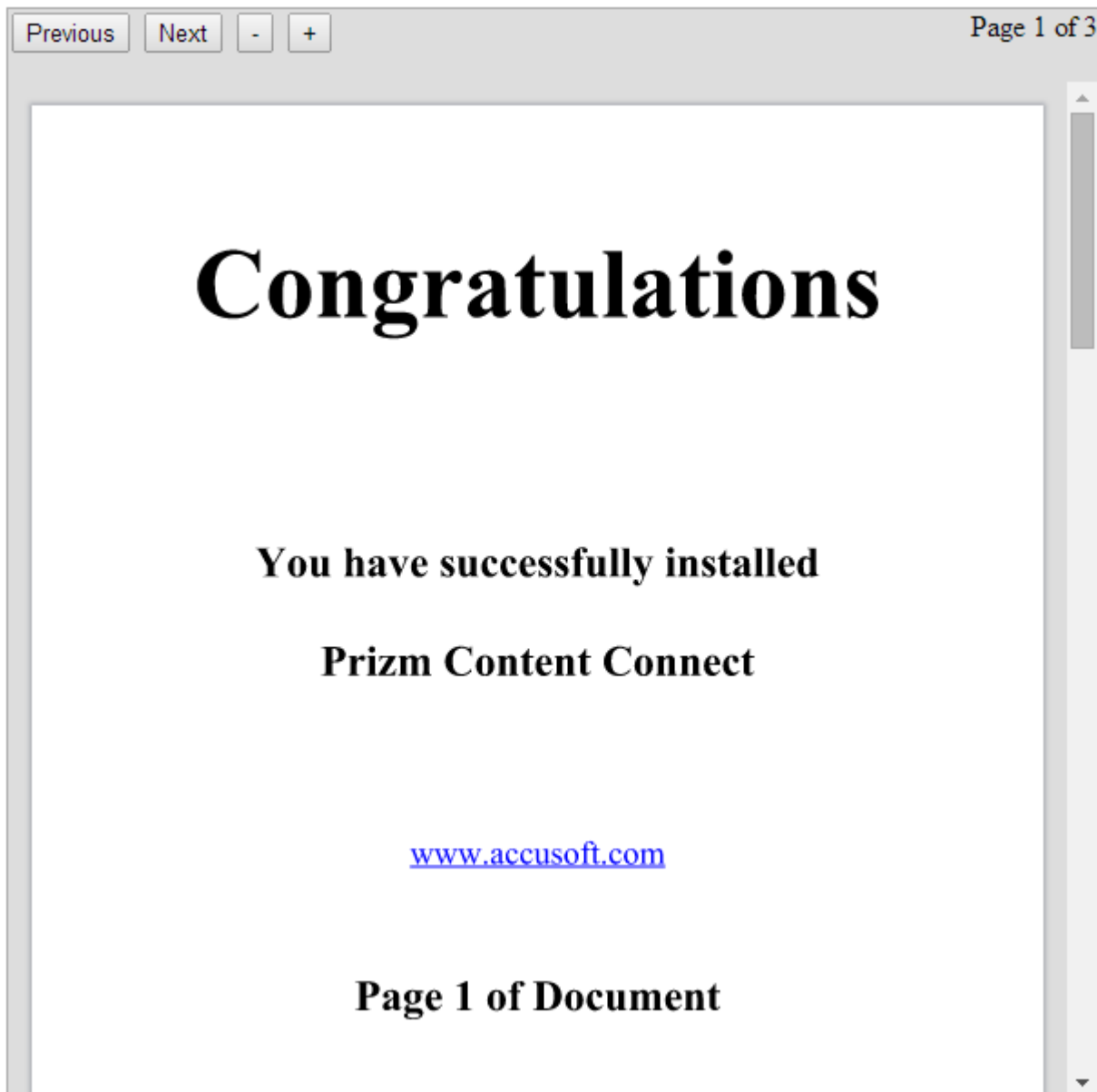
It is possible to build a custom Viewer UI by directly embedding the [ViewerControl](#) instead of embedding the Viewer using the jQuery plugin.

There are several benefits to building a custom Viewer UI, including the ability to:

- Design menus and button placement that is unique for your customer's workflow.
- Create custom UI elements/behavior.
- Arrange UI elements distributed throughout a web page, rather than all Viewer UI elements in a single div.
- Choose your own UI framework(s).
 - The ViewerControl (viewercontrol.js) does not have a dependency on any third-party frameworks/libraries

Customization Example

In the example below, the [ViewerControl](#) is embedded into a page and a simple UI is built around the [ViewerControl](#). The Viewer created by the example code is shown in the figure below:



The directory structure for this example is:

- /
 - css/
 - viewercontrol.css (product)
 - simple.css (custom - shown below)
 - js/
 - viewercontrol.js (product)
 - viewerCustomizations.js (product)
 - simple.js (custom - shown below)
 - simple.html (custom - shown below)

NOTE: This example uses jQuery for simplicity and because it is well known to many readers. jQuery is served from the Google Hosted Libraries CDN.

HTML (simple.html)

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title>Custom Viewer UI Example</title>
  <!--resources for the viewer-->
  <link rel="stylesheet" href="css/viewercontrol.css">
  <script src="js/viewercontrol.js"></script>
  <script src="js/viewerCustomizations.js"></script>
  <!--Use jQuery to build the UI of the viewer-->
  <script src="//ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">
</script>
  <!-- Script tag for simple.js, which creates and embeds the custom viewer. -->
  <script src="js/simple.js"></script>
  <!-- CSS for the page -->
  <link rel="stylesheet" href="css/simple.css">
</head>
<body>
<!-- This div contains the custom viewer and UI elements. -->
<div class="viewerWrapper">
  <!-- There are some buttons/tools in the viewer UI. These will be disabled
until the ViewerReady event. -->
  <button id="prevPage" class="viewerButton" disabled>Previous</button>
  <button id="nextPage" class="viewerButton" disabled>Next</button>
  <button id="zoomOut" class="viewerButton" disabled>-</button>
  <button id="zoomIn" class="viewerButton" disabled>+</button>
  <!-- There is a status bar that shows the current and total page number. -->
  <div class="floatRight">Page <span id="currentPage">1</span> of <span
id="totalPages">1</span></div>
  <!-- The ViewerControl is embedded in this element. -->
  <div id="viewerControlContainer">ViewerControl goes here</div>
</div>
</body>
</html>
```

JavaScript (js/simple.js)

```
$(document).ready(function() {
  // Get the element where the viewer will be embedded.
```

```
var element = document.getElementById("viewerControlContainer");
// Create the options object for the viewer
var options = {
  documentID : "a_valid_document_id",
  imageHandlerUrl: "my_pas_proxy",
  icons : viewerCustomizations.icons,
  language : viewerCustomizations.languages['en-US'],
  // printTemplate : "...", // include a print template for printing
to work
};
// Create the viewer control
var viewerControl = new PCCViewer.ViewerControl(element, options);
// It's best practice to wait for the "ViewerReady" event before calling the
viewer API.
viewerControl.on("ViewerReady", function() {
  // Display the page count when the estimated and actual page count are
available.
  viewerControl.on("PageCountReady", displayPageCount);
  viewerControl.on("EstimatedPageCountReady", displayPageCount);
  function displayPageCount(ev) {
    $("#totalPages").html(ev.pageCount);
  }
  // Display the current page number when the page changes.
  viewerControl.on("PageChanged", function(ev) {
    $("#currentPage").html(ev.pageNumber);
  });
  // It's safe to enable the UI buttons during the ViewerReady event
$("#viewerButton").prop("disabled", false);
// ...and then hookup the UI buttons
$("#nextPage").click(function() {
  viewerControl.changeToNextPage();
});
$("#prevPage").click(function() {
  viewerControl.changeToPrevPage();
});
$("#zoomOut").click(function() {
  viewerControl.zoomOut(1.25); // Zoom out 1.25x
});
$("#zoomIn").click(function() {
  viewerControl.zoomIn(1.25); // Zoom in 1.25x
});
});
});
```

NOTE: The sample code that installs with the product demonstrates how to generate a document ID. The code is too large to be shown in this example.

CSS (css/simple.css)

```
.viewerWrapper {
  width: 600px;
  height: 600px;
  position: relative;
  border: 1px solid #aaa;
  background: #ddd;
}
#viewerControlContainer {
  position: absolute;
```

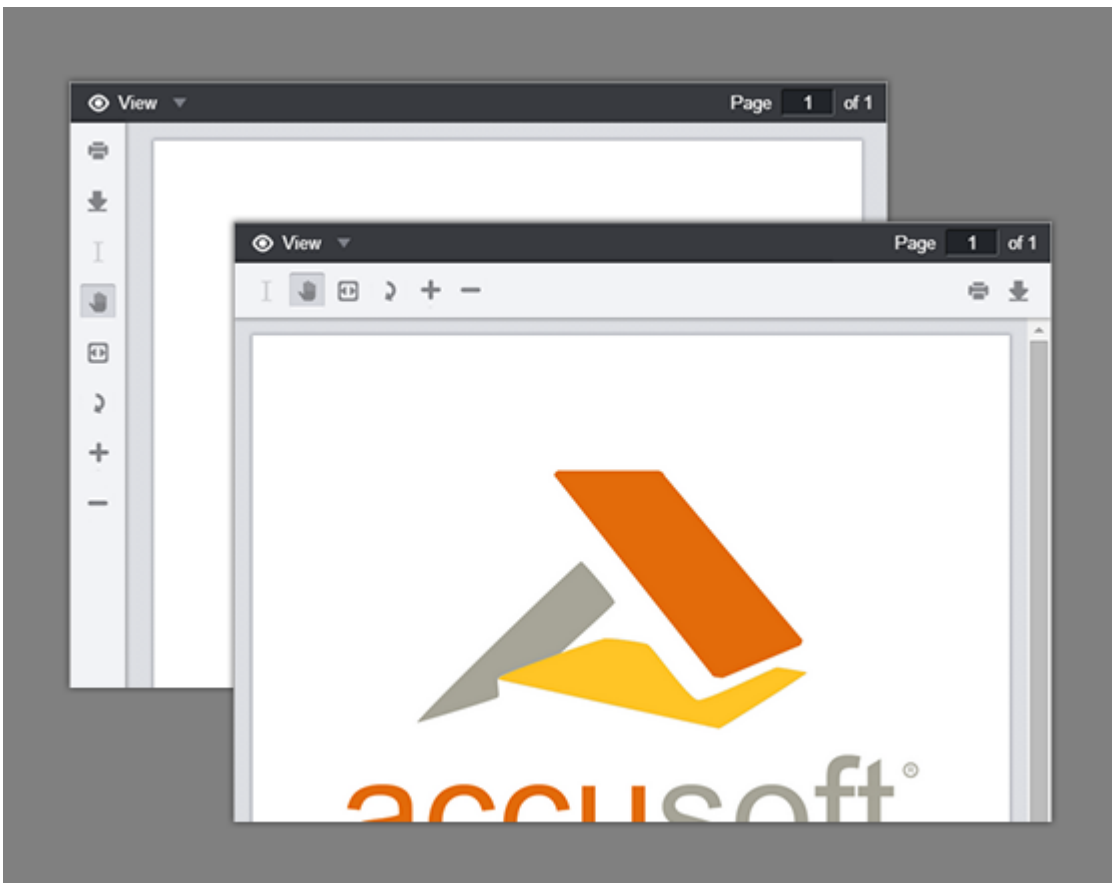
```
top: 40px;
bottom: 0px;
left: 0px;
right: 0px;
}
.floatRight {
float: right
}
```

Change the Position of the Menu Bar

Introduction

You can change the layout of the tab panes by modifying the markup located in the **viewerTemplate.html** file by:

- Adding the **pcc-tab-vertical** class to make the tab pane vertical.
- Adding either a **pcc-left** or **pcc-right** class to specify the side on which the vertical tab pane will appear.



Example

```
<div class=\"tabset pcc-nav-tabset\" data-pcc-nav>
  <!-- Tab -->
  <div class=\"pcc-tab\" data-pcc-nav-tab=\"demo\">
    <div class=\"pcc-tab-item\">Demo</div>
    <!-- This tab pane is vertical and left aligned -->
```

```
<div class\="pcc-tab-pane pcc-tab-vertical pcc-left"\>
  <!-- Tab content -->
</div>
</div>
<!-- End tab -->
</div>
```

Create a Custom Mouse Tool

Introduction

You can create a custom mouse tool in the Viewer by defining the tool and then by updating the UI to show the button for the tool.

Step 1: Define the tool in JavaScript code

Define the tool in the code before the tool is selected. For example:

Example

```
// Create the new mouse tool. var myTool = PCCViewer.MouseTools.createMouseTool(
    "PinkLine",
    PCCViewer.MouseTool.Type.LineAnnotation);

// Configure the tool to draw a pink (#FF69B4) line that is 10 pixel thick
myTool.getTemplateMark()
    .setColor("#FF69B4")
    .setThickness(10);
```

Step 2: Update the UI to show a button for the tool

This modification will take place in the **viewerTemplate.html** file. The button can be added to several places in the UI, but a common place to add the button is in the Annotate tab pane. Content of the Annotate tab is defined in the element with the attribute **data-pcc-nav-tab="annotate"**:

Example

```
<!-- The following markup will create a button that enables use
of the mouse tool named "PinkLine".

The custom attributes that are used:
  \* data-pcc-mouse-tool="PinkLine" - specifies that the button selects the
mouse tool named "MyLineTool"
  \* data-pcc-context-menu="false" - specifies that a context menu is not shown
for this mouse tool
-->
<button
  data-pcc-mouse-tool="PinkLine"
  data-pcc-context-menu="false"
  class\="pcc-icon pcc-icon-annotate-line"
```

```
title="Pink Line Tool"></button>
```

Create a Custom Tab

Introduction

You can add a new custom tab to the Viewer tab navigation by modifying the markup located in the **viewerTemplate.html** file. The tab must be placed inside the `div` element with the data attribute `data-pcc-nav` and it must have a unique `data-pcc-nav-tab` value:

Example

```
<div class="pcc-tabset pcc-nav-tabset" data-pcc-nav>
  <!-- An example of a custom tab -->
  <div class="pcc-tab" data-pcc-nav-tab="custom">
    <div class="pcc-tab-item">
      <!-- Tab label -->
    </div>
    <div class="pcc-tab-pane">
      <!-- Tab content -->
    </div>
  </div>
  <!-- End custom tab -->
</div>
```

Customize the Markup

Introduction

All of the markup in the Viewer is customizable. This topic covers how to customize using the templates, custom attributes, and the mouse tools.

Templates

You can change the markup of the Viewer UI components by editing the templates. The templates are HTML files ending in ***.Template.html**. The primary navigation tabs and menus are located in the **viewerTemplate.html** file.

Adding/Removing Template Files

To create your own version of the Viewer template you will need to update the template name where it is being loaded in `viewer.js`. For example, if you copy **viewerTemplate.html** to a new file called **customTemplate.html** you would change:

Example

```
element.html(\_.template(options.template.viewer, ...
```


To:

Example

```
element.html(\_.template(options.template.custom, ...
```

Currently, in the samples "**Template**" is removed from the name of the template property in the template object. For example, **viewerTemplate.html** becomes **template.viewer**.

Template Syntax

The templates are consumed using the **Underscore.js Template** utility function. Variables and JavaScript conditions can be used within the templates using ERB syntax. For more information, refer to the Underscore documentation: <http://underscorejs.org/#template>.

Example

```
<!-- An example of a variable -->
<button data-pcc-rotate class\="pcc-icon pcc-icon-rotate"
        title="\<%= rotate %>"\></button>
```

Custom Attributes

Throughout the templates, on many elements, there are data attributes starting with **data-pcc-**. These are used to identify elements and bind them to functionality defined in the **viewer.js** file:

Example

```
<!-- This button will rotate the current page when clicked -->
<button data-pcc-rotate class\="pcc-icon pcc-icon-rotate"\></button>

<!-- Other elements can perform the same function -->
<div data-pcc-rotate class\="customClass"\></div>
```

Using the example above, this attribute is used as a selector for a **\$rotatePage** jquery object in **viewer.js**:

Example

```
this.viewerNodes = {
  $rotatePage: viewer.$dom.find("\[data-pcc-rotate\]") ...
```

This attribute is then bound to a click event which rotates the current page:

Example

```
// Rotate Page button viewer.viewerNodes.$rotatePage.on('click', function () {  
    viewer.viewerControl.rotatePage(90);  
});
```

Mouse Tools

The named mouse tools, like this one, are provided by **viewercontrol.js**:

Example

```
<button  
    data-pcc-mouse-tool="AccusoftSelectToZoom"  
    class=\"pcc-icon pcc-icon-rectanglezoomtool\"></button>
```

Using a Custom Mouse Tool in the Viewer UI

A named mouse tool can be used in the default UI of the Viewer. This allows one or more tools to be pre-configured. Setting the `data-pcc-mouse-tool` attribute to false will prevent the context menu from opening if the mouse tool is an annotation or redaction.

Enabling a custom tool in the Viewer UI requires modification of the **viewerTemplate.html** file, as shown in the example below:

Example

```
<!-- The following markup will create a button that enables use  
    of the mouse tool named "MyLineTool". -->  
<button  
    data-pcc-mouse-tool="MyLineTool"  
    data-pcc-context-menu="false"  
    class=\"pcc-icon pcc-icon-annotate-line"  
    title="My Line Tool\"></button>
```

Customize the Mouse Tools

Mouse tools are named, customizable instances that can be used by any Viewer instance. Each mouse tool has a type, which determines how the tool behaves and the properties the tool has.

Mouse Tool Names

Mouse tools are given a name when a mouse tool is created. This name is used to get and set the mouse tool that is used by the Viewer:

Example

```
// The name of a new mouse tool  
var myMouseToolName = "MyLineTool";
```

```
// Create the new mouse tool
PCCViewer.MouseTools.createMouseTool(
    myMouseToolName,
    PCCViewer.MouseTool.Type.LineAnnotation);

// Set the current mouse tool of the ViewerControl by passing the name
viewerControl.setCurrentMouseTool(myMouseToolName);
```

The mouse tool name also specifies the new mouse tool in the **MouseToolChanged** event:

Example

```
// The MouseToolChanged event triggers when the mouse tool changed.
// Use the ViewerControl#getCurrentMouseTool() method or event#mouseToolName
property
// to get the new mouse tool name.
viewerControl.on("MouseToolChanged", function(ev) {
    viewerControl.getCurrentMouseTool() == ev.mouseToolName; // true
});
```

Mouse Tool Objects

PCCViewer.MouseTool objects represent mouse tools, and the objects can be used to configure the mouse tools.

These objects are returned when creating a mouse tool with method

PCCViewer.MouseTools.createMouseTool(...). The object can be retrieved at a later time using the method **PCCViewer.MouseTools.getMouseTool()**:

Example

```
// Get the MouseTool object for an existing tool
var myMouseTool = PCCViewer.MouseTools.getMouseTool(myMouseToolName);
```

The **MouseTool** object has getters for the tool name and type:

Example

```
myMouseTool.getName(); // returns "MyLineTool"
myMouseTool.getType(); // returns "LineAnnotation"
```

Depending on the tool type, additional getters or setters may be available to configure the tool:

Example

```
// All mouse tools that draw a mark (annotation and redaction) have a
// getter `getTemplateMark()`
if (myMouseTool.getType() === "LineAnnotation") {
    // The method gives access to a template mark that configures how the tool
```

```
draw
  // the annotation or redaction.
  // In this example the mouse tool is configured to draw a red line.
  myMouseTool.getTemplateMark().setColor("#FF0000");
}
```

Mouse Tool Type

- The mouse tool type specifies the behavior of a mouse tool.
- The API has many different mouse tool types, all of which are specified in the enumeration **PCCViewer.MouseTool.Type**.
- The mouse tool type is specified when creating the mouse tool, and it cannot be changed.

Multiple Mouse Tools of one Type

Creating multiple mouse tools of one type is useful if several pre-defined behaviors are needed for one type of mouse tool. For example, this gives the ability to create two text highlighter tools, one that highlights red and the other that highlights green:

Example

```
// Create the red highlighter
PCCViewer.MouseTools.createMouseTool(
  "RedHighlighter",
  PCCViewer.MouseTool.Type.HighlightAnnotation)
  .getTemplateMark()
  .setFillColor("#FF0000");

// Create the green highlighter
PCCViewer.MouseTools.createMouseTool(
  "GreenHighlighter",
  PCCViewer.MouseTool.Type.HighlightAnnotation)
  .getTemplateMark()
  .setFillColor("#00FF00");
```

Pre-defined Named Mouse Tools

The file **viewercontrol.js** creates several named mouse tools as listed in the table below. These named mouse tools are used by the Viewer out-of-the-box.

Name	Type
"AccusoftMagnifier"	PCCViewer.MouseTool.Type.Magnifier
"AccusoftSelectToZoom"	PCCViewer.MouseTool.Type.SelectToZoom
"AccusoftPan"	PCCViewer.MouseTool.Type.Pan
"AccusoftPanAndEdit"	PCCViewer.MouseTool.Type.PanAndEdit
"AccusoftSelectText"	PCCViewer.MouseTool.Type.SelectText
"AccusoftEditMarks"	PCCViewer.MouseTool.Type.EditMarks

Name	Type
"AccusoftLineAnnotation"	PCCViewer.MouseTool.Type.LineAnnotation
"AccusoftArrowAnnotation"	PCCViewer.MouseTool.Type.LineAnnotation
"AccusoftRectangleAnnotation"	PCCViewer.MouseTool.Type.RectangleAnnotation
"AccusoftEllipseAnnotation"	PCCViewer.MouseTool.Type.EllipseAnnotation
"AccusoftTextAnnotation"	PCCViewer.MouseTool.Type.TextAnnotation
"AccusoftStampAnnotation"	PCCViewer.MouseTool.Type.StampAnnotation
"AccusoftHighlightAnnotation"	PCCViewer.MouseTool.Type.HighlightAnnotation
"AccusoftRectangleRedaction"	PCCViewer.MouseTool.Type.RectangleRedaction
"AccusoftTransparentRectangleRedaction"	PCCViewer.MouseTool.Type.TransparentRectangleRedaction
"AccusoftTextRedaction"	PCCViewer.MouseTool.Type.TextRedaction
"AccusoftStampRedaction"	PCCViewer.MouseTool.Type.StampRedaction

Customize the Styles

Introduction

This topic covers how to customize the styles in the Viewer. Depending on your needs, you can make minor changes to the Viewer by following the guidance below. Or, if you want fine-grained control of the Viewer's look and feel, you can go to the [PrizmDoc Viewer Client Assets Build Guide](#) for detailed instructions.

Customizing the Styles

Before you begin, make sure that the included .css file is loaded in the Viewer: normalize.min.css.

Namespace

The Viewer uses the class **.pccv** to namespace the styles it uses. In order to override any selector used in the Viewer, your selector must begin with the class **.pccv**:

Example

```
/* Set the navigation tab bar to dark red */
.pccv .pcc-nav-tabset,
.pccv .pcc-nav-tabset .pcc-tab-item,
.pccv .pcc-status-bar { background: #5b100d; }
```

Organization

All resulting CSS files have a Less counterpart in the root of the **less** folder. These are the only files that can be built on their own. File names beginning with an underscore (**_**) are partial style files, and are included as modules in the root files. These individual components are split out into the following structure:

- **base** - These files contain the variables and mixins used by the Viewer, as well as the overall layout. Included

here are also the reusable, generic components, such as the grid and form inputs.

- **components** - These files contain the large Viewer components, and are named in a self-explanatory way. For example, styles related to the search functionality are held in the **_search.less** file.

Variables

There are many variables contained in `less/base/_variables.less`, which control things like the image resources, color scheme, and toolbar sizing. These variables can be modified in order to propagate changes throughout the Viewer.

Icons

A number of icons are used throughout the Viewer for different UI elements. These icons are stored in the `icons\svg*.svg` files. Any individual SVG file can be scaled up or down for use at any viewport size. Additionally, the color of any SVG file can be altered via CSS attributes for use on both light and dark backgrounds.

Media Queries

The Viewer utilizes CSS3 Media Queries with expressions using *min-width* and *max-width* to adjust the layout of navigation and dialogs. The Media Query Breakpoints are set according to the Viewer layout. On smaller viewports the tab navigation collapses into a menu and some tools are hidden. On larger viewports the dialogs transform from horizontal to a vertical layout to utilize screen real estate.

The breakpoints are located in the `less/base/_breakpoints.less` file, and are used throughout the less files as detached rulesets. The breakpoints are as follows:

Example

```
/* Target modern browsers that support media queries */
.modernView(@rules) {
  @media (min-width: 0) { @rules(); }
}

/* Mobile & Tablet Sizes, collapse navigation tabs into menu */
.mobileView(@rules) {
  @media (max-width: 767px) { @rules(); }
}

/* Desktop Sizes */
.desktopView(@rules) {
  @media (min-width: 768px) { @rules(); }
}
```

Grid System

The Viewer uses a basic grid system to assist with the UI layout. Through a series of rows and columns the layout can scale dynamically. Rows are used to create horizontal groups of columns. Columns are created by defining the number of twelve columns you will span. For example, three columns would use three divs with a class of **.pcc-col-4**:

Example

```
<div class="pcc-row">
  <div class="pcc-col-4">Left</div>
  <div class="pcc-col-4">Center</div>
  <div class="pcc-col-4">Right</div>
```

```
</div>
```

Legacy CSS support

The legacy CSS necessary for legacy browsers is held in the **less/legacy.less** file. Here we address unsupported or troublesome CSS features like drop shadows, opacity or background alpha transparency. In addition, because Media Queries are not supported in some legacy browsers and no Media Query polyfills are used in this regard, we add additional styles here that are accounted for in Media Queries in modern browsers.

Polyfills

There are a few polyfills used to provide support for modern browser features:

- **Normalize** (<https://necolas.github.io/normalize.css/>) - Normalize provides better cross-browser consistency in the default styling of HTML elements.

Disable the Print Button

Introduction

There are two ways of disabling printing inside the Viewer:

1. Just disable printing when initializing the viewer
2. Make a custom build of the viewer which has the printing button removed from the UI

Option 1: Just Disable Printing When Initializing the Viewer

You can easily disable printing when initializing the viewer. Just add the following `uiElements` parameter to the `options object`:

Example

```
<script type="text/javascript">
  $(function() {
    $('#viewerContainer').pccViewer({
      /*
       ...other required options omitted for clarity...
      */
      uiElements: {
        printing: false
      }
    });
  });
</script>
```

Option 2: Make a Custom Build of the Viewer

If you need more customization of the viewer UI than the initialization API allows, you can always perform your own custom build of the viewer after modifying the original HTML templates. To do this, you will need the source code for the client viewer build, available at <https://github.com/Accusoft/prizmdoc-viewer>.

If you wanted to remove the printing button from the UI in this way, you would need to modify `src/templates/viewerTemplate.html`, removing the element with the attribute `data-pcc-print="launch"`.

After you have made your changes, perform your own build and use the new `dist/viewer-assets` output in place of the existing `viewer-assets` in your application.

For more information about building the client viewer, see the [README](#).

Additional Resources

- [Design Basics](#) - This topic gives you an overview of how to customize the Viewer.
- [Available Parameters for the UI](#) - This topic provides a list of available UI configuration options.
- [Interactive Code Examples](#) - These live code examples allow you to see how to customize the Viewer.

Enable Multiple Redaction Reasons

Introduction

When creating a redaction in the PrizmDoc Viewer Client UI, you can apply multiple redaction reasons to be associated with the selected redaction. These reasons will be visible in the Viewer and saved to PDF along with the rest of the redacted content.

How to Enable Multiple Redaction Reasons

The Viewer uses the single redaction reason mode by default. You need to explicitly enable it to use Multiple Redaction Reasons.

When [creating the viewer](#) define the redaction reasons list and enable multiple redaction reasons with a JavaScript call like this:

```
<script type="text/javascript">
  $(function() {
    $('#viewerContainer').pccViewer({
      ...
      annotationsMode: "LayeredAnnotations",
      redactionReasons: {
        autoApplyDefaultReason: true,
        enableRedactionReasonSelection: true,
        enableFreeformRedactionReasons: true,
        maxLengthFreeformRedactionReasons: 40,
        enableMultipleRedactionReasons: true,
        reasons: [
          {
            reason: "1.a",
            description: "Client Privilege"
          },
          {
            reason: "1.b",
            description: "Privacy Information due to HIPAA regulations"
          },
          {
            reason: "1.c",
            description: "Redacted"
          }
        ]
      }
    })
  })
}
```

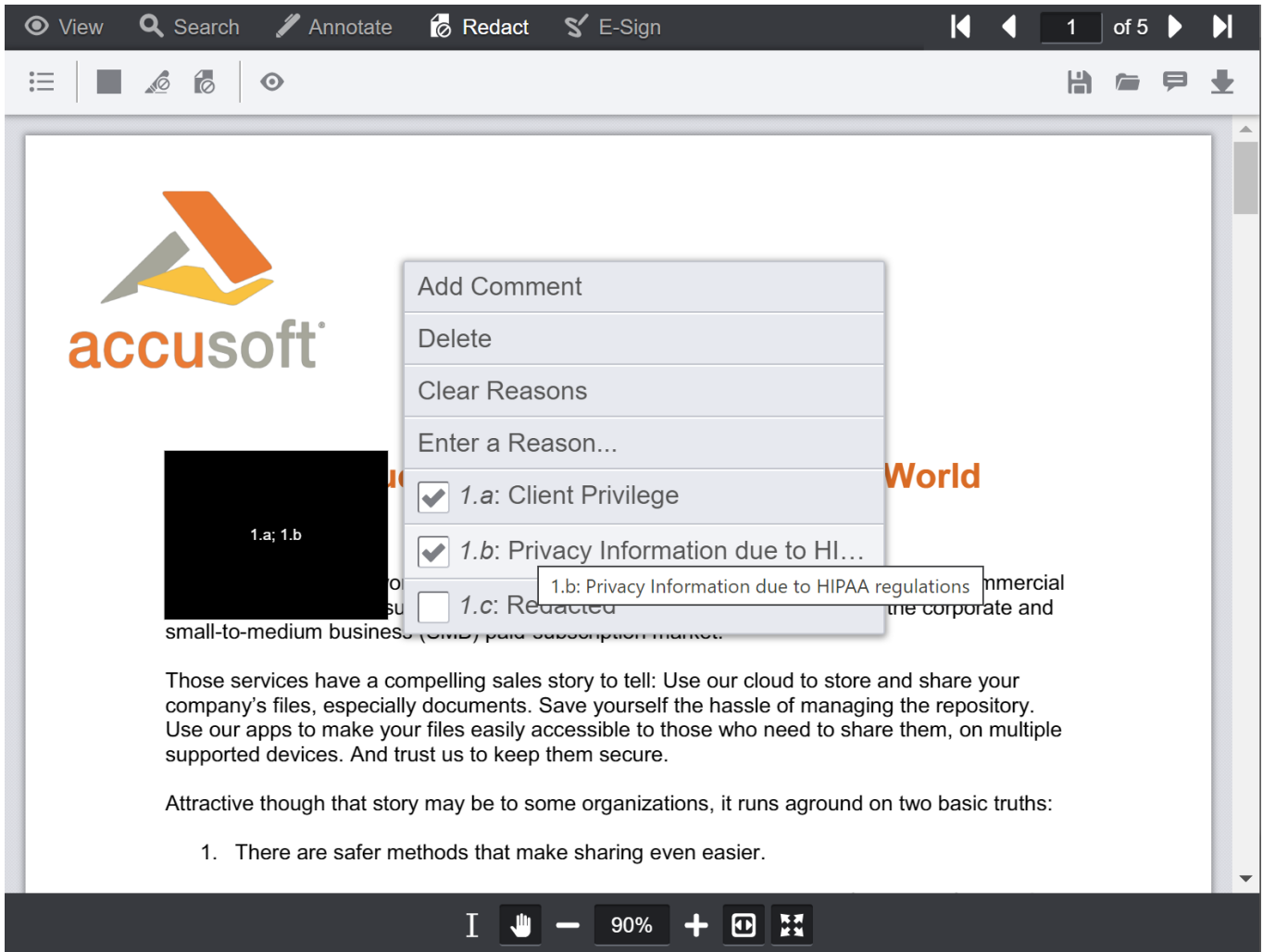


```

    }
  });
});
</script>

```

If you run your web application now and create some redactions, the Viewer should display the reasons list with descriptions and allow you to select multiple reasons:



For a complete list of `redactionReasons` options when initializing the Viewer plugin, refer to the [redactionReasons API](#) topic.

How to hide and show action items in the immediate menu

Let's say we don't want the "Add Comment" and "Delete" items to be shown in the immediate menu. It can be configured with the `immediateActionMenuActionsFilter` property from viewer [Options](#):

```

<script type="text/javascript">
  $(function() {
    $('#viewerContainer').pccViewer({
      ...
      annotationsMode: "LayeredAnnotations",
      redactionReasons: {
        ...

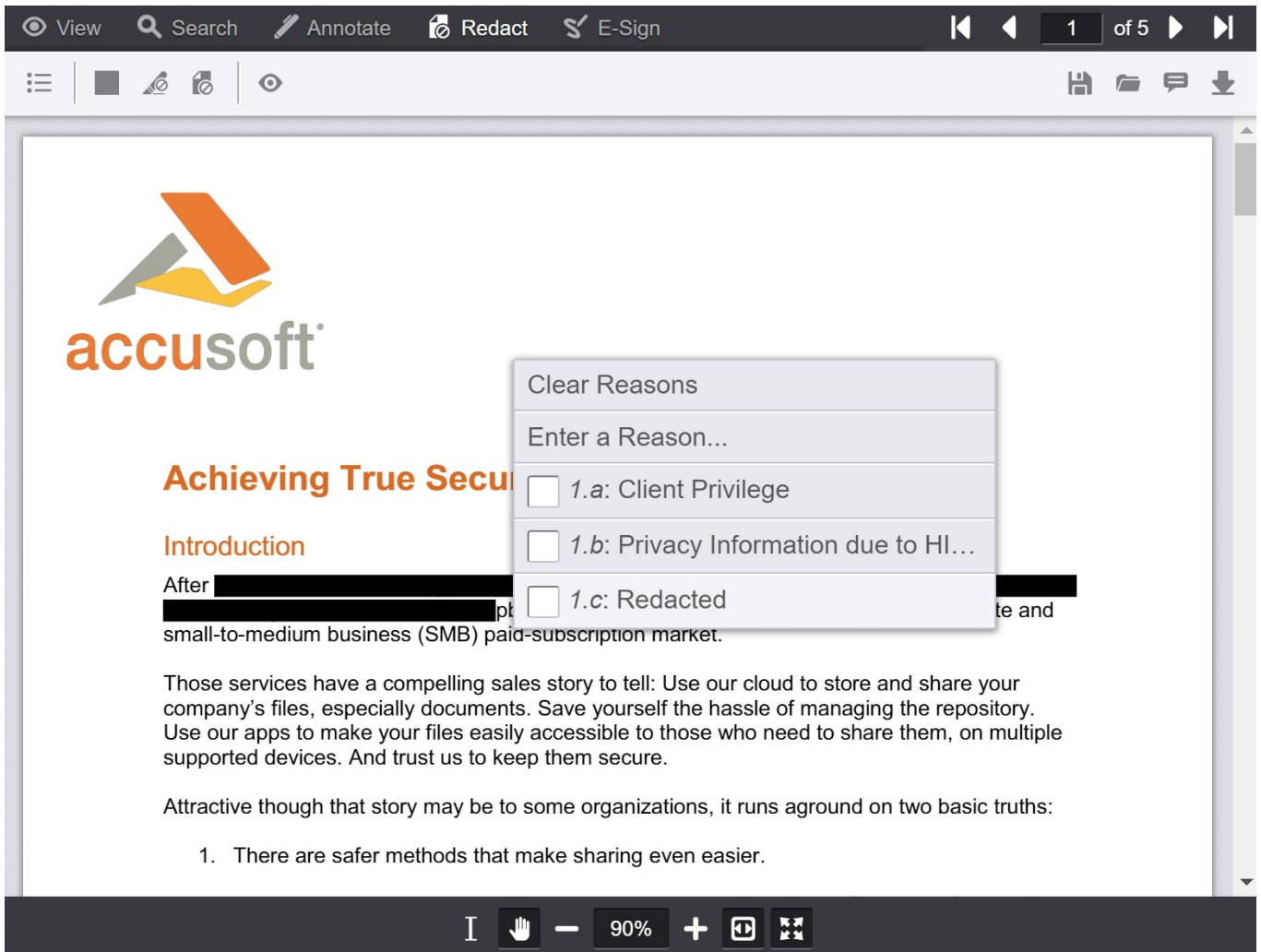
```

```

    },
    immediateActionMenuActionsFilter: {
      comment: false,
      "delete": false
    }
  });
});
</script>

```

The immediate menu will look like the following after this change:



How to adjust immediate menu size

You can adjust the immediate menu by changing the CSS style for the `pcc-immediate-action-menu` class. Create a file with the name `viewer-style-customization.css` and put content:

```

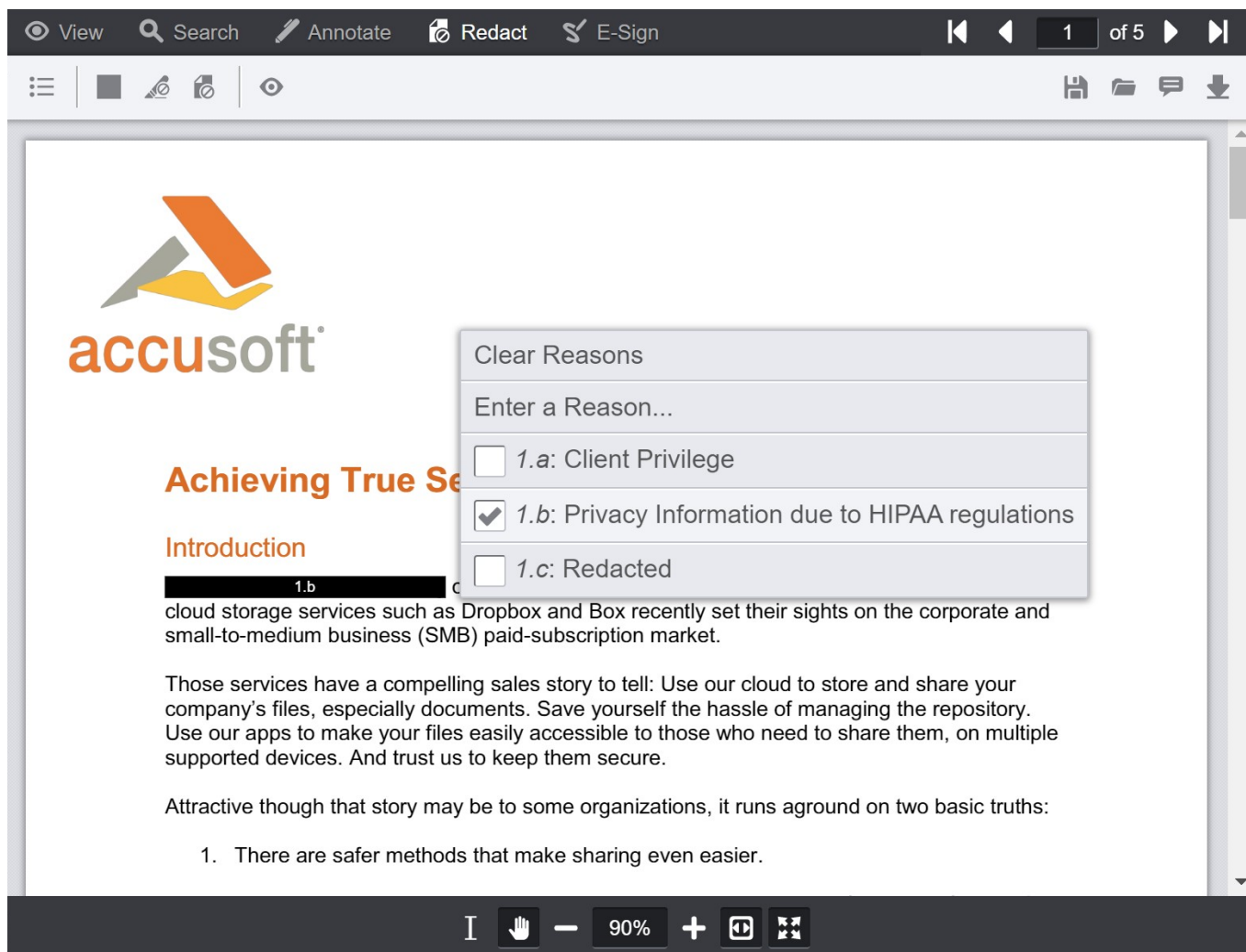
.pcc-immediate-action-menu {
  max-width: 450px;
  max-height: 75%;
}

```

Then update your application HTML file and add a link to the `viewer-style-customization.css` file so that it is placed after the `viewer-assets/css/viewer.css` link:

```
<link rel="stylesheet" href="viewer-assets/css/viewer.css">
<link rel="stylesheet" href="viewer-style-customization.css">
```

It will result in the following view:



Reorganize Menus

All of the menus and navigation in the Viewer are customizable.

Templates

You can change the markup of the Viewer UI components by editing the templates. The templates are HTML files ending in ***.Template.html**. The primary navigation tabs and menus are located in **viewerTemplate.html**.

Adding/Removing Template Files

If you wish to create your own version of the Viewer template, you will need to update the template name where it

is being loaded in viewer.js. For instance, if you copy **viewerTemplate.html** to a new file called **customTemplate.html** you would change:

Example

```
element.html(_.template(options.template.viewer, ...
```

To:

Example

```
element.html(_.template(options.template.custom, ...
```

Currently in the samples, "Template" is removed from the name of the template property in the template object (e.g., **viewerTemplate.html** becomes **template.viewer**).

Template Syntax

The templates are consumed using the Underscore.js Template utility function. Variables and JavaScript conditions can be used within the templates using ERB syntax. For more information, see the Underscore documentation at <http://underscorejs.org/#template>.

Example

```
<!-- An example of a variable -->  
<button data-pcc-rotate class="pcc-icon pcc-icon-rotate"  
  title="<%= rotate %>"></button>
```

Removing Elements

To remove buttons from the menu, you can remove the markup, comment out the markup, or add a CSS class to the element.

Example

```
<!-- This button is no longer visible  
  <button data-pcc-rotate class="pcc-icon pcc-icon-rotate"></button>  
-->  
  
<!-- This element is hidden because of a pcc-hide class -->  
<div data-pcc-rotate class="customClass pcc-hide"></div>
```

Data Attributes

Throughout the templates, on many elements, there are data attributes starting with **data-pcc-**. These are used to identify elements and bind them to functionality defined in viewer.js:

Example

```
<!-- This button will rotate the current page when clicked -->
<button data-pcc-rotate class="pcc-icon pcc-icon-rotate"></button>

<!-- Other elements can perform the same function -->
<div data-pcc-rotate class="customClass"></div>
```

Customizing CSS

It is recommended to create your own CSS file and add it to the bottom of the cascade. This would be after **normalize.min.css**, **viewer.css**, **viewercontrol.css**, and **legacy.css**. All selectors in **viewer.css** have a parent of **.pccv**, in order to override the **.pccv** parent should be in the selector:

Example

```
/* Set the navigation tab bar to dark red */
.pccv .pcc-nav-tabset,
.pccv .pcc-nav-tabset .pcc-tab-item,
.pccv .pcc-status-bar {background:#5b100d}
```

CSS Polyfills

There are a few polyfills used to provide support for modern browser features:

- **Normalize** (<http://necolas.github.io/normalize.css/>) - Normalize provides better cross-browser consistency in the default styling of HTML elements.

Scroll the Viewer Programmatically

The ViewerControl API offers two methods for scrolling the Viewer:

- The 'scrollToAsync(target)' method will scroll the Viewer to a specified target/location within the Viewer.
- The 'scrollBy(offsetX, offsetY)' method will scroll the Viewer by a specified number of pixels.

Scroll to a Specific Target

The API enables scrolling to a specific target or location within the Viewer. Possible targets are:

- a mark (annotation, redaction, or signature)
- a conversation
- a search result, or
- a point on a specific page

Use the method `PCCViewer.ViewerControl#scrollToAsync(target)` to scroll to a specific target, as shown in the following code example:

Example

```
myViewerControl
  .scrollToAsync({
    pageNumber: 2,
    x: 500,
    y: 500
  })
  .then(
    function onFulfilled() {
      alert("The point was scrolled into view.")
    },
    function onRejected() {
      alert("Something went wrong: ")
    }
  );
```

The `scrollToAsync` method returns a `PCCViewer.Promise` that is fulfilled when the target is scrolled into view and has been displayed. In the `onFulfilled` callback is an appropriate time to take additional programmatic actions against the target (e.g. put a mark in text editing mode).

The method will attempt to center the target. If the full target does not fit at the current scale, then it will attempt to center the top left corner of the target. The method will not scroll past the bounds of the document, so in some cases it will scroll to as close to centering the target as possible.

There is also the `ViewerControl#scrollTo(target)` method that returns the `ViewerControl` object rather than a promise. It is recommended that you use `scrollToAsync`, unless method chaining is desired.

Scroll by a Number of Pixels

The API enables scrolling the content in the Viewer by a specified number of pixels. The method `PCCViewer.ViewerControl#scrollBy(offsetX, offsetY)` allows scrolling up, down, left, or right. Scrolling down or right is performed by passing positive offset values. Scrolling up or left is performed by passing negative offset values.

The following example demonstrates how to scroll down using this method:

Example

```
// Scroll down by 100 pixels
myViewerControl.scrollBy(0, 100);
```

This method will scroll content until it reaches the bounds of the document, at which point the method will stop scrolling in that direction.

When the `ViewerControl` is in `SinglePage` view mode, this method will only scroll within the current page, it will not change pages.

Set the Initial Zoom Factor

Example Integration

When a document loads, it defaults to the size of the screen. While this ensures all documents fit on the screen, certain documents may be sized too large. To prevent this, you can set an initial zoom of the document before it is displayed. To set a specific scaling factor before the document loads with optimal performance, subscribe to the

PageOpening event to set the target scale before PageDisplayed has fired.

The PageOpening event is triggered when a page being opened has reached the point that it has height, width, and resolution data but hasn't yet been displayed. This example sets the initial zoom of the Viewer at runtime so that it will initialize at 125% zoom factor instead of auto-fit:

Example

```
function pageOpeningHandler() {
    // Set the initial page zoom to a maximum of 125%
    if (viewerControl.getScaleFactor() > 1.25) {
        viewerControl.setScaleFactor(1.25);
    }
    // Remove the event subscription since we only want to respond to the first
    PageOpening
    viewerControl.off(PCCViewer.EventType.PageOpening, pageOpeningHandler);
}
viewerControl.on(PCCViewer.EventType.PageOpening, pageOpeningHandler);
```

Subscribe to Events

Subscribing to ViewerControl Events

The [ViewerControl](#) has several events defined in the enumeration `PCCViewer.EventType` that can be subscribed to using the `.on` method. The `.off` method can be used to unsubscribe to previously subscribed events. Event subscription allows you to provide a function[m1] that is called whenever the event is fired. This method is called a 'handler'. When the event is triggered, the handler is called with one argument, a [PCCViewer.Event](#) object that is augmented with properties specific to the event type.

Event subscription should be performed immediately after the [ViewerControl](#) is created, or within the `ViewerReady` event, to ensure that no events are triggered before subscription.

Example

```
var viewer = new PCCViewer.ViewerControl(viewerElement,
    {
        documentID: viewingSessionId,
        imageHandlerUrl: "../pcc.ashx"
    });

viewer.on(PCCViewer.EventType.ViewerReady, viewerReadyHandler);

function viewerReadyHandler(event) {
    alert("The ViewerReady event was fired.");
    // Now Subscribe to the event PCCViewer.EventType.PageChanged
    //exposed by the API
    viewer.on(PCCViewer.EventType.PageChanged, function(event) {
        alert("The current Page is " + event.pageNumber);
    });
    //subscribe to other events here...
}
```

An enumeration, **PCCViewer.EventType**, defines event types that are triggered by the [ViewerControl](#).

It's acceptable to use this enum or use string literals when subscribing and unsubscribing.

Work with Annotations

This section provides details on how to change annotation default values, load annotations from a web tier, work with annotation layers and work with annotations programmatically:

- [Change Annotation Default Values](#)
- [Load Annotations from the Web Tier](#)
- [Work with Annotation Layers](#)
- [Work with Annotations Programmatically](#)

Change Annotation Default Values

Changing Annotation Default Values

You can customize the default behavior of an annotation by modifying the button markup located in **viewerTemplate.html**.

In this example the mouse tool will draw a green line and will not open the context menu:

Example

```
<button data-pcc-mouse-tool="AccusoftLineAnnotation"
  data-pcc-default-fill-color="#ff0000"
  data-pcc-context-menu="false"></button>
```

Default Data Attributes

These are the current data attributes that are being used to set defaults:

Attribute	Value Type	Description
data-pcc-default-fill-color	String	The default fill color for the annotation. This must be a hexadecimal string of 6 characters preceded by a #.
data-pcc-context-menu	Boolean	Whether or not to show the context menu when the annotation is drawn. Default is true.

Load Annotations from the Web Tier

Annotation layers may be persisted to the web tier and then loaded back into the Viewer at a later time. On the web tier, the resource representing a layer is referred to as the 'markup layer record' while in the Viewer, it's referred to as the 'markup layer object'. An important distinction to make is that the layer object in the Viewer has two IDs associated with it while a layer record persisted to the web tier has only one. A layer object in the Viewer has both an 'id' and a 'markupLayerRecordId' while a persisted layer record has only a 'markupLayerRecordId'.

What's the difference? A layer's 'id' is a runtime identifier that exists only for the life of the layer object. This 'id' is needed for several reasons; one of them being as a unique object identifier for the period of time when a layer is

created in the Viewer but not yet persisted to the web tier. Only when a layer is persisted to the web tier will it have a 'markupLayerRecordId'.

The following is a code example for loading layer records in to the Viewer:

Example

```
// Get a ViewerControl object
var viewer = new PCCViewer.ViewerControl(viewerElement, {
    documentID: viewingSessionId,
    imageHandlerUrl: "../pcc.ashx"
});

// Retrieve a list of the annotation layer records persisted on the web tier.
viewer.requestMarkupLayerNames().then(
    function onResolve(annotationLayerRecords) {
        // Load a specific record so that its data is available to the API
        viewer.loadMarkupLayers(annotationLayerRecords[0].layerRecordId).then(
            function onResolve(annotationLayers){
                // A layer object representing the persisted record is now created.
                // Any marks and their associated comments are now displayed
                // on the loaded document.
                console.log("Annotation layer loaded: ", annotationLayers[0]);
            },
            function onReject(reason) {
                console.log("Failed to load annotation layer. ", reason);
            }
        );
    },
    function onReject(reason) {
        console.log("Failed to load annotation layer record list. ", reason);
    }
);
```

Work with Annotation Layers

PrizmDoc Viewer's Annotation Layering functionality makes it possible to create, view and manage multiple sets of annotations for a document enabling collaborative annotating and commenting scenarios for your document review needs.

What's an Annotation Layer?

An annotation layer (or layer) is a collection of annotations saved in a unique file (under this definition all annotation files you currently have are layers). The layer may be modified over time as often as desired and can be modified by different users, however, only a single user can work with the file at a time.

In order to maintain layer integrity, only one user can be modifying a layer file at any given moment.

The best way to achieve this is by having each user create their own layer for editing; this is the main scenario that annotation layers are designed upon. Note that there is nothing in the PrizmDoc Viewer code that will restrict the incorrect usage of layer files, so this is an important consideration for your implementation.

Review Layers

A "Review Layer" is simply a layer that has been loaded but cannot be modified by the current user. The user may load as many layers for review as they like, and each of these layers can be made independently visible/invisible via the user interface. While the user cannot modify any of the annotations on a Review Layer, they may comment on them which makes it possible to have conversations across multiple reviewers of a document.

With this system it's possible to have as many people working concurrently to annotate a document as desired, and these people can see all other annotation layers if they wish (the layers will be as up-to-date as the last time the file was loaded for a user).

Annotation File Format

When adding the Annotation Layering functionality, we updated our annotation storage mechanism to accommodate cross-layer references for commenting. For this reason, we have moved to a JSON file format for all annotation persistence. This JSON format is based on our ability to persist individual annotations into a JSON object.

It's important to note that once annotation layering has been turned 'on', via the Viewer configuration option, that all persistence will use the JSON file format. This means that when an XML file (legacy) is loaded for editing, saving it again will save it as a JSON file. When both an XML and JSON file exist with the same root filename, only the JSON file will be visible to the user (it will be as if the XML file no longer exists, although it still remains as a separate file).

For more detailed information, refer to the following topics:

- [Loading Annotations from the Web Tier](#)
- [API - jQuery.fn](#)

Work with Annotations Programmatically

Introduction

The API offers methods to perform create, read, update, and delete operations on annotations. Other API methods available are for selection/de-selection of marks, obtaining selected annotations, re-ordering of annotation mark objects (within a z-order of a page), and loading and saving of annotations.

Create & Update Annotations

When adding annotations programmatically using the **addMark method** to distant pages of a large document, or if you are not sure if your annotation object will straddle the page width and height boundaries, it is recommended that you use the **requestPageAttributes method** to obtain width and height of a page. This will help you to remain within the confines of the image rectangle when specifying width and height of the annotation's dimensions (rectangle or start point and end point).

Example

```
var pageNumber = 1;
var promise = viewer1.requestPageAttributes(1);
promise.then(
    function(pageAttributes) {
        try{
```

```
        //create stamp annotation on page 1
        var stampMark1 = viewer1.addMark(1,
PCCViewer.Mark.Type.StampAnnotation);
        //update the created rectangle mark using the property setters
        //use width and height obtained in the promise object to place the
stamp annotation object
        // assume the width obtained was 612 and height 792.
        stampMark1.setRectangle({x: 250, y: 50, width :
            pageAttributes.width - 300, height: pageAttributes.height -
500});

        //provide the label for the stamp mark
        stampMark1.setLabel("Reviewed");
        //set the color to red
        stampMark1.setColor("#ff0000");
        // ... add more annotations on this page 1 ...
        var rectangleMark1 = viewer1.addMark(1,
PCCViewer.Mark.Type.RectangleAnnotation);
        rectangleMark1.setRectangle({x: 250, y: 50, width :
            pageAttributes.width - 300, height: pageAttributes.height - 500});
        //set fill color to blue
        rectangleMark1.setFillColor("#0000ff");
    }
    catch(e){
        alert("ERROR: " + e);
    }
},
function(rejectedReason) {
    alert("Unable to add annotations because page attributes promise was
rejected, error = " +
        rejectedReason);
}
);
```

Create Text-Based Mark from Search Result

You can create a text-based mark from a search result by using [addMarkFromSearchResult](#).

Read Annotations Properties

Property getters can be used to obtain current property values. It is not necessary to use `requestPageAttributes` method for these operations.

Example

```
var label = stampMark1.getLabel();
var rectangle = stampMark1.getRectangle();
```

Select Annotations

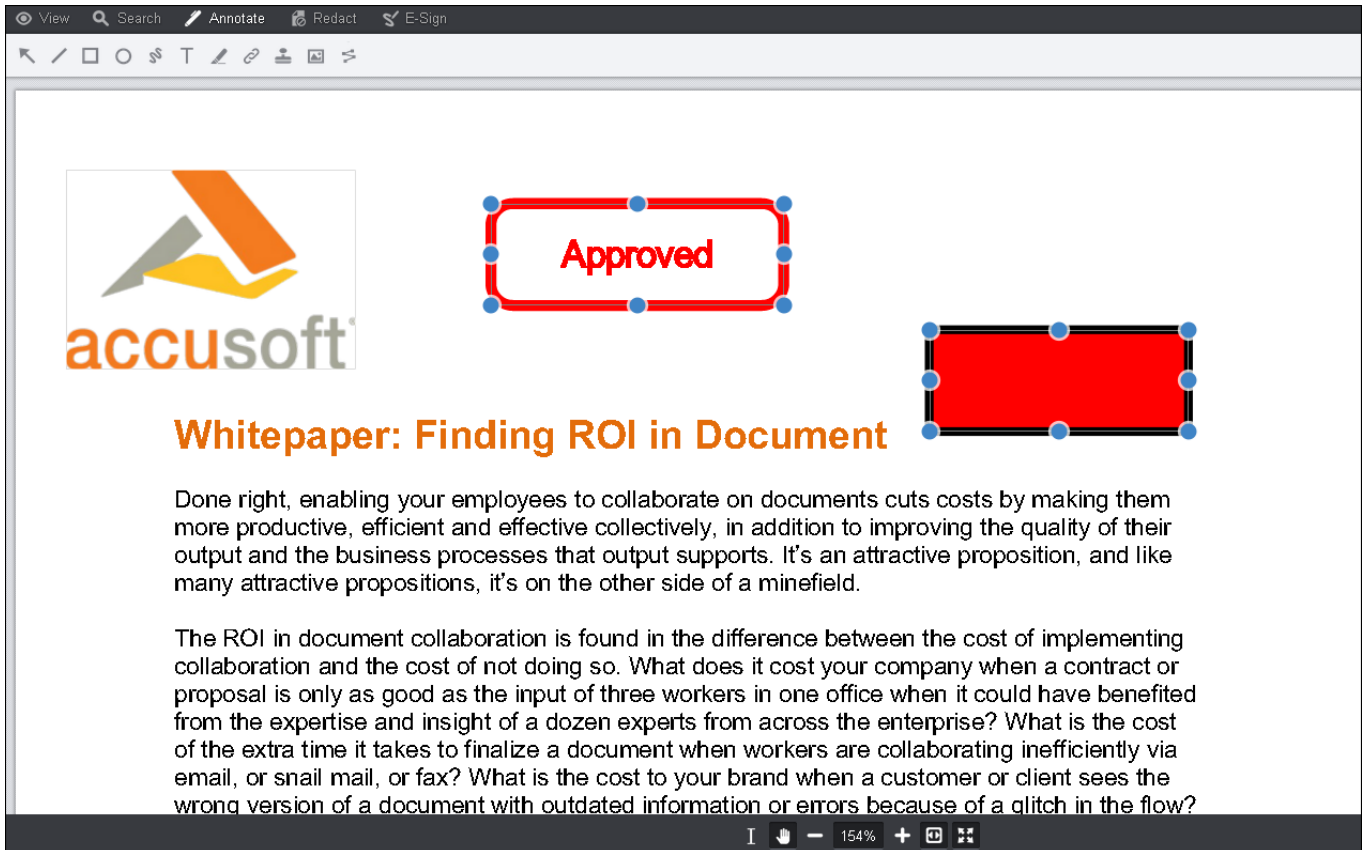
The following example shows how to select annotations:

Example

```
//get all marks var marks = getAllMarks();

//select annotations returned by the above getAllMarks() call (marks is an array)
viewer1.selectMarks(marks);
```

The graphic below shows the previously created marks are selected after selectMarks call:



Deselect Annotations

The following example shows how to deselect annotations:

Example

```
//Get the selected marks var selectedMarks = viewer1.getSelectedMarks();
if(selectedMarks.length > 0 ) {
    //deselect all the marks in all the pages
    viewer1.deselectMarks();
    //programmatically check if all marks were deselected
    var checkSelectedMarks = viewer1.getSelectedMarks();
    if(checkSelectedMarks.length === 0 ) {
        alert("All annotations marks were deselected");
    }
}
```

Reorder Annotations

Reordering of marks applies to all marks on a single page. Reordering does not apply to Highlight annotations because these type of marks need to remain attached to the Text on the page.

Example

```
//use moveMarkToFront to bring mark to the front
viewer1.moveMarkToFront(rectangleMark1);

//use moveMarkToBack to the back viewer1.moveMarkToFront(stampMark1);

//use moveMarkForward to bring the mark one slot forward
viewer1.moveMarkForward(rectangleMark1);

//use moveMarkBackward to move the mark one slot backward
viewer1.moveMarkBackward(rectangleMark1);
```

Delete Annotations

The following example shows how to delete annotations:

Example

```
//In this example we will delete all the marks that are selected
//Get the selected marks var selectedMarks = viewer1.getSelectedMarks();
//delete if there were any selected if(selectedMarks.length > 0 ) {
    //delete all the selected marks
    viewer1.deleteMarks(selectedMarks);
}
```

Save Annotations

You can save annotations created in the currently displayed document using [saveMarkupLayer](#).

Load Annotations

Refer to the topic, [Load Annotations from the Web Tier](#).

Work with Document Comparison Programmatically

Introduction

The API offers methods to request revisions and to get those requested revisions once they have been retrieved.

Requesting and Getting Revisions

Request the revisions for a given document comparison. Revisions requests complete asynchronously. A [PCCViewer.RevisionsRequest](#) will be returned that provides events for revisions retrieval progress and members to

access retrieved revisions:

Example

```
// Request revisions
var revisionsRequest = viewerControl.requestRevisions();
// Subscribe to the PartialRevisionsAvailable event to get revisions as they become
available
revisionsRequest.on('PartialRevisionsAvailable', function(_event) {
    // Get the newly available revisions
    var newRevisions = _event.partialRevisions;
});
// Subscribe to the RevisionsRetrievalCompleted event to get revisions when all of
them are available
revisionsRequest.on("RevisionsRetrievalCompleted", function(){
    // Get all of the revisions for that comparison
    var revisions = revisionsRequest.getRevisions();
});
```

Modify viewer.js

Introduction

To facilitate open customization of the Viewer, this file is left unminified and unobfuscated. This file controls the behavior of the user interface, allowing you to bind custom button behavior or to completely re-implement the user experience to match any business need. This file utilizes the public [ViewerControl API](#) to allow complete interaction with the underlying Page List document control. To find out more about this, consult the [ViewerControl API](#) section.

Updates to the PrizmDoc Viewer Product

This file will be updated with future releases of the product, introducing new features and enhancing the current behavior, when necessary. When editing or re-implementing this file, a clear upgrade path should be established in order to be able to take full advantage of future releases of the product.

Viewer.js Sections

The file is split up into several logical sections, in order to make modifying the file easier. They are as follows, in order:

Using the Viewer Template and Parsing for DOM Elements

The Viewer Template is inserted into the specified Viewer element, and then individual components are parsed out using jQuery. For convenience, all jQuery-wrapped elements are placed in a variable starting with the **\\$** character. For example, the pan tool button element is named **\\$panTool**, indicating that the object has the full jQuery API available. This naming is maintained throughout the file.

Initialization and Binding the Markup

After the DOM elements are parsed out, behavior is bound to them inside a single initialization function, named **bindMarkup**. The content of the short initialization function can be seen right above, in **initializeViewer**. All DOM

behavior, such as click and input events, are bound to the DOM here using the jQuery API.

Auxiliary Functions

Following are some auxiliary functions, which provide useful and reusable abstractions and wrappers for the [ViewerControl API](#). An example of this is the **setMouseTool** function, allowing any part of the file to set the mouse tool through the [ViewerControl API](#) and update all necessary DOM elements accordingly. Other functions include handling toggle elements, displaying notifications in the Viewer, handling context menu and dialog behavior, and various others.

ViewerControl Event Handlers

Next are all of the event handler functions. These are written in separate functions in order to allow easy subscribe and unsubscribe handling. These include events that are currently handled in the Viewer. For a list of all available events, consult the **EventType** section of the [ViewerControl API](#).

Create the ViewerControl and Add Listeners

Directly following the event handler, the main Page List viewer control is initialized, and the necessary events are subscribed. Initializing the main control requires only the `\$pageList` DOM element, and the original **options** object passed into the Viewer and jQuery plugin. For more specifics on initializing the Viewer control, consult the [ViewerControl API](#).

The DOM element passed into the Viewer control constructor should be a plain DOM element, and not the jQuery variable. This is why `\$pageList.get(0)` is used in the code.

Search and Annotation IO Modules

Next are two sections that have been abstracted in a module format. The first handles the search navigation UI, and the second handles retrieving, opening, and saving annotations. These modules are self-contained, as much as possible, to allow easy removal of these large parts of code if you are not interested in that specific functionality.

jQuery Plugin

This is the first part of code being executed, and is a simple jQuery wrapper, providing convenience for the Viewer. It will create a new instance of the Viewer, and can also provide the **ViewerControl** instance associated to a Viewer in a particular DOM element. For more information on this plugin, consult the **jQuery Plugin** section of the [ViewerControl API](#).

Legacy Samples

Introduction

If you are self-hosting your PrizmDoc Viewer backend, we have some legacy samples which you may find useful (these samples assume you are self-hosting the backend and are not currently designed to be easily configured with our cloud-hosted backend options). These legacy samples are included in the "Client Installer".

The legacy samples each demonstrate a variety product functionality within a single web application. These samples are available for the following server-side frameworks:

- [ASP.NET MVC](#)
- [ASP.NET WebForms](#)
- [JSP](#)

Click on any of the links above for instructions on how to setup the sample web application.

Legacy ASP.NET MVC Sample

Installation

1. Ensure that Microsoft's Internet Information Service (IIS) is enabled on the computer that will be running the .NET MVC 5 sample. For steps on how to enable IIS, go to the [How to Enable Internet Information Services](#) page.
2. Ensure that .NET 4.5 is installed. You can download it from the [Microsoft .NET Framework](#) page.
3. Run the "Client Installer".
4. During installation, make sure the following features are selected to be installed:
 - o Legacy Samples
 - o PAS (PrizmDoc Application Services)
 - o Configure ASP.NET Samples with IIS
 - o Re-register ASP.net 4.0 with IIS

5. After installation, open this URL to make sure the legacy sample is running:

`http://localhost:18000/PrizmDoc_HTML5_Viewer_NET_MVC`

Overview

1. From the splash page you have three options:

Choice of viewer:

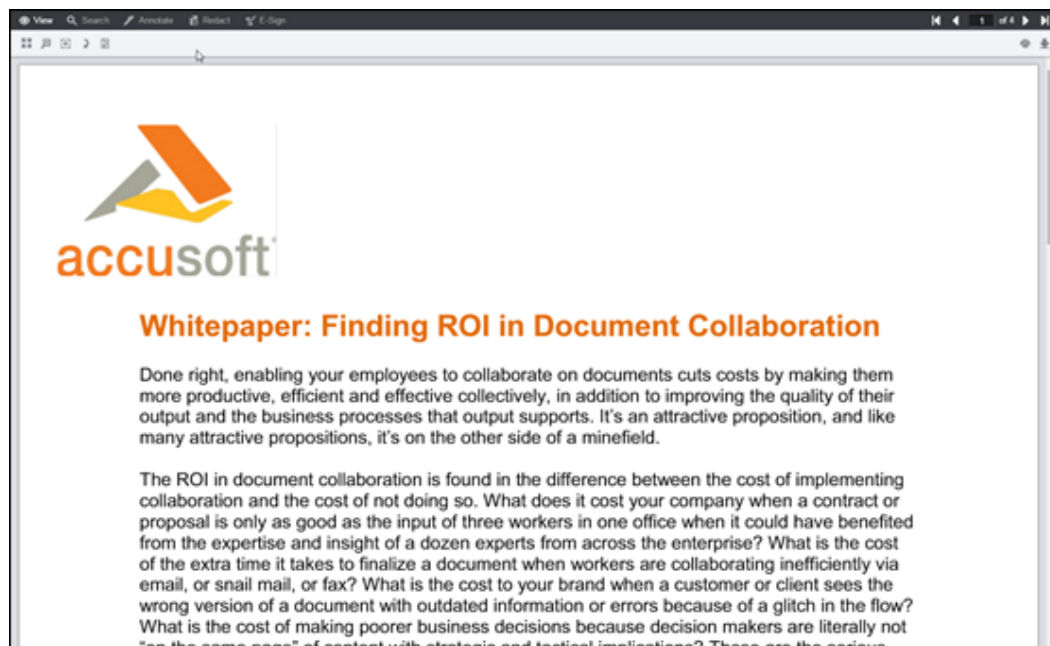
- o You can choose to load either the **Full Viewer**, **Book Reader** or the **Comparison Viewer**.

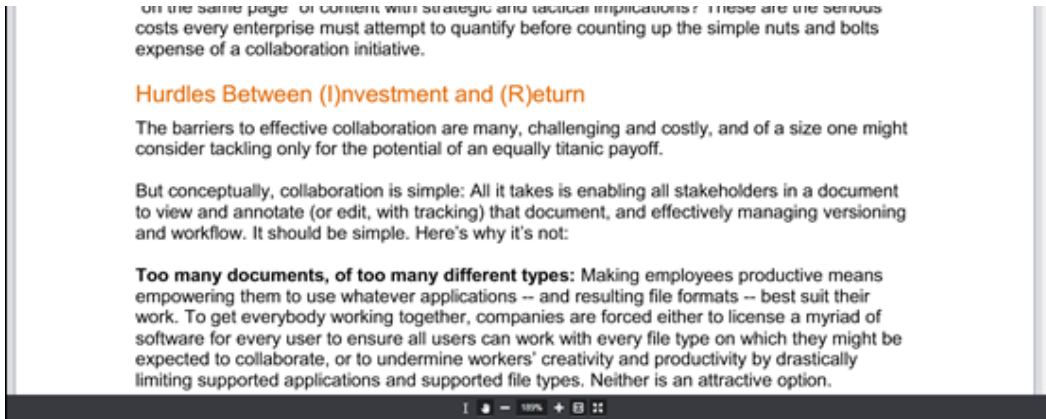
Select a sample document -OR- upload a document:

- o You can choose any of the 5 sample documents (Word, PDF, CAD, Tiff, or JPEG)
- o Or, you can upload a document from an arbitrary location on your computer. Note that dragging and dropping a file on this page is not supported in Internet Explorer 8.

2. Full Viewer:

If you select **Full Viewer** on the splash page, then documents will be viewed with the full-featured, out-of-the-box Viewer:





3. Book Reader:

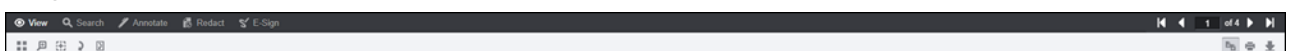
If you select **Book Reader** on the splash page, then documents will be viewed with the book reader. The book reader demonstrates how the Viewer can be heavily customized:

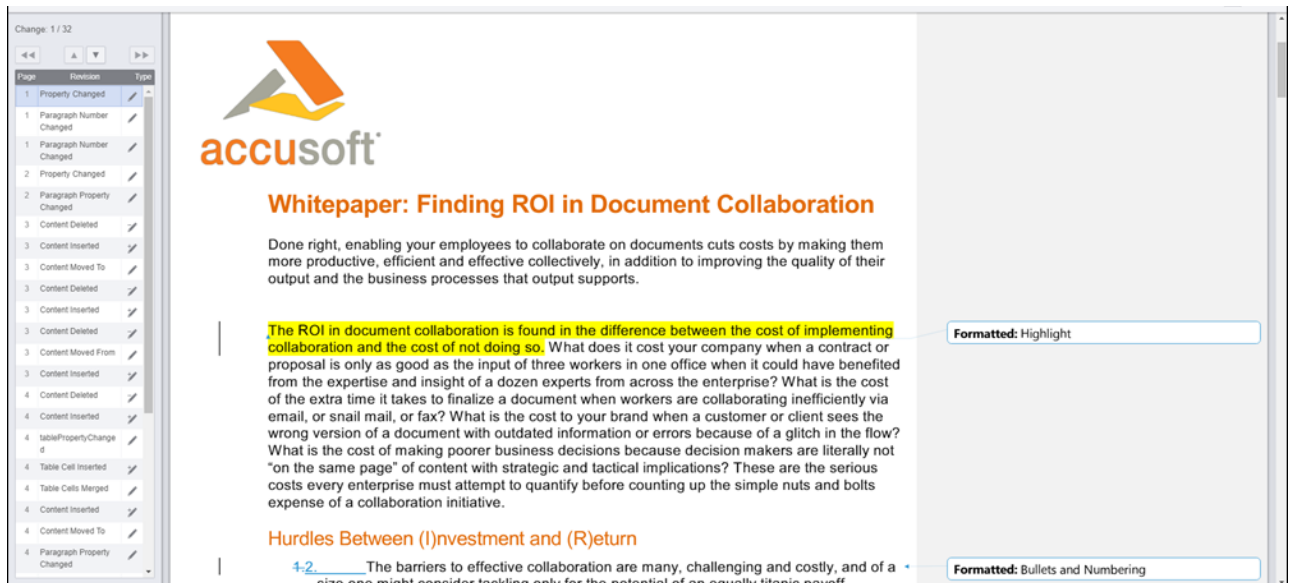


4. Comparison Viewer

NOTE: This feature requires a [Microsoft Office enabled PrizmDoc Viewer License](#).

If you select **Comparison Viewer** on the splash page, then you can upload two Word (.doc or .docx) files to be compared in the Viewer. The Comparison Viewer shows how two Word documents can be reviewed and compared using PrizmDoc Viewer:





Sample Directory Structure

The legacy ASP.NET MVC sample is installed at `C:\prizm\Samples\dotnet\mvc`. This folder contains all the MVC related folders (Models, Views and Controllers), all the Visual Studio related files and our different viewers which are located on the **viewers** folder.

PccViewerServices Route

In `App_Start/RouteConfig.cs` you will find one special route called `PccViewerServices`. This route will catch all requests made to the application that start with `pcc/`. The `{*pathInfo}` fragment is very important as it will be needed by our Controller later on.

PccController

The `PccController` handles the requests from the route in the previous section. It simply passes the `pathInfo` information to our own route handler which will handle the request appropriately.

NOTE: If you are interested in seeing how we handle the requests, please take a look at the source code in `Modes/PccViewer`.

Folder contents: viewers/full-viewer-sample

NOTE: The full viewer (with the `comparisonMode` configuration parameter set to `true`) is used for document comparison.

File / Folder	Description
<code>viewer-assets</code> folder	Contains the essential JavaScript, CSS, fonts, and images that make up the Viewer.
<code>viewer-assets/src</code> folder	Contains Less, icons, languages, and HTML templates that can be used to build the Viewer CSS and customizations. This folder is non-essential, and does not need to be re-distributed.
<code>viewer-assets/Gulpfile.js</code>	Contains Gulp tasks to build the Viewer Less, icons, and HTML templates. This file is non-essential and does not need to be re-distributed.
<code>viewer-</code>	A file used by npm (a package manager). It defines the dependencies installed by npm, which

File / Folder	Description
<code>assets/package.json</code>	are required to run Gulp and compile the Viewer Less, icons, and HTML templates. This file is non-essential and does not need to be re-distributed.
<code>predefinedSearch.json</code>	This data file contains information defining search queries that will appear as selectable items in the full viewer. > NOTE: This file is consumed by the page <code>Default.aspx</code> and the JSON is injected into the HTML that is returned by <code>Default.aspx</code> . Ultimately, the predefined search terms are provided as a JavaScript hash, when the Viewer is created.
<code>redactionReason.json</code>	This data file contains information defining redaction reasons that are available in the Viewer. > NOTE: This file is consumed by the page <code>Default.aspx</code> and the JSON is injected into the HTML that is returned by <code>Default.aspx</code> . Ultimately, the redaction reasons are provided as a JavaScript hash, when the Viewer is created.

Folder contents: `viewers/book-reader-sample`

File / Folder	Description
<code>viewer-assets</code> folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the book reader viewer.
<code>viewer-assets/less</code> folder	Contains less that can be used to build the book reader CSS. This folder is non-essential, and does not need to be re-distributed.
<code>viewer-assets/Gruntfile</code>	Contains Grunt tasks to build the reader less. This file is non-essential and does not need to be re-distributed.
<code>viewer-assets/package.json</code>	A file used by npm (a package manager). It defines the dependencies installed by npm, which are required to run Grunt and compile the less.
<code>viewer-assets/selection.json</code>	A file used by the IcoMoon application to generate the icons in the book reader viewer. If you need to add an icon to the Viewer, you can add the icon to this file and use the IcoMoon application (https://icomoon.io) to generate a new icon font. This file is non-essential and does not need to be re-distributed.
<code>viewer-assets/js/sample-config.js</code>	Contains references to the assets, web tier, and language files used by the Viewer in this sample.

Folder contents: `viewers/e-signer-sample`

File / Folder	Description
<code>modules</code> folder	Contains uncompiled assets of the Viewer. These files will be compiled to <code>viewer-assets/js/bundle.js</code> and <code>viewer-assets/css/bundle.css</code> by the build process defined in <code>Gulpfile.js</code> . The files in this folder are non-essential and do not need to be re-distributed.
<code>viewer-assets</code> folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the Viewer.
<code>Gulpfile.js</code>	Contains Gulp tasks to build the viewer js and css files. This file is non-essential and does not need to be re-distributed.
<code>viewer-assets/package.json</code>	A file used by npm (a package manager). It defines the dependencies installed by npm, which are required to run Gulp and compile the Viewer assets. This file is non-essential and does not need to be re-distributed.
<code>index.html</code>	The default page for the sample. This page calls the <code>pcc.ashx</code> handler to start a viewing session

File / Folder	Description
	with PAS and then the page loads the Viewer.
<code>webpack.config.js</code>	Webpack configuration file. This file contains all the settings for the webpack module bundler. We use webpack to compile all the files in the modules folder to the <code>bundle.js</code> and <code>bundle.css</code> that are found in the <code>viewer-assets</code> folder.

Folder contents: `viewers/template-designer-sample`

File / Folder	Description
modules folder	Contains uncompiled assets of the Viewer. These files will be compiled to <code>viewer-assets/js/bundle.js</code> and <code>viewer-assets/css/bundle.css</code> by the build process defined in <code>Gulpfile.js</code> . The files in this folder are non-essential and do not need to be re-distributed.
<code>viewer-assets</code> folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the Viewer.
<code>Gulpfile.js</code>	Contains Gulp tasks to build the viewer js and css files. This file is non-essential and does not need to be re-distributed.
<code>viewer-assets/package.json</code>	A file used by npm (a package manager). It defines the dependencies installed by npm, which are required to run Gulp and compile the Viewer assets. This file is non-essential and does not need to be re-distributed.
<code>index.html</code>	The default page for the sample. This page calls the <code>pcc.ashx</code> handler to start a viewing session with PAS and then the page loads the Viewer.
<code>webpack.config.js</code>	Webpack configuration file. This file contains all the settings for the webpack module bundler. We use webpack to compile all the files in the modules folder to the <code>bundle.js</code> and <code>bundle.css</code> that are found in the <code>viewer-assets</code> folder.

Configuration with `pcc.config`

The file `pcc.config` is used to configure the resources and storage used by the Viewer web tier. This file can be found in the root of the sample folder. This file is self-documenting, but a little information about the configuration options is given below.

Option	Description
<code><DocumentPath></code>	The sample pulls named documents from this location. The <code>DocumentPath</code> must have read/write permissions in order for the file drag and drop functionality of the splash page to work.
<code><PrizmApplicationServicesScheme></code>	Specifies the scheme (<code>http</code> or <code>https</code>) to use when connecting with PAS.
<code><PrizmApplicationServicesHost></code>	Specifies the host to use when connecting with PAS.
<code><PrizmApplicationServicesPort></code>	Specifies the port to use when connecting with PAS.

Development Information

The legacy ASP.NET MVC sample has the following requirements for development:

- Visual Studio 2012 or later
- .NET 4.5 or later

Legacy ASP.NET WebForms Sample

Installation

1. Prior to installation, ensure Microsoft's Internet Information Service (IIS) and ASP.NET 4.0+ are enabled on the computer that will be running the .NET Web Forms sample.
2. Run the "[Client Installer](#)".
3. During installation, make sure the following features are selected to be installed:
 - o **Legacy Samples**
 - o **PAS (PrizmDoc Application Services)**
 - o **Configure ASP.NET Samples with IIS**
 - o **Re-register ASP.net 4.0 with IIS**

4. After installation, open this URL to make sure the legacy sample is running:

`http://localhost:18000/PrizmDoc_HTML5_Viewer_NET_WEBFORMS`

Overview

1. From the splash page you have three options:

Choice of Viewer:

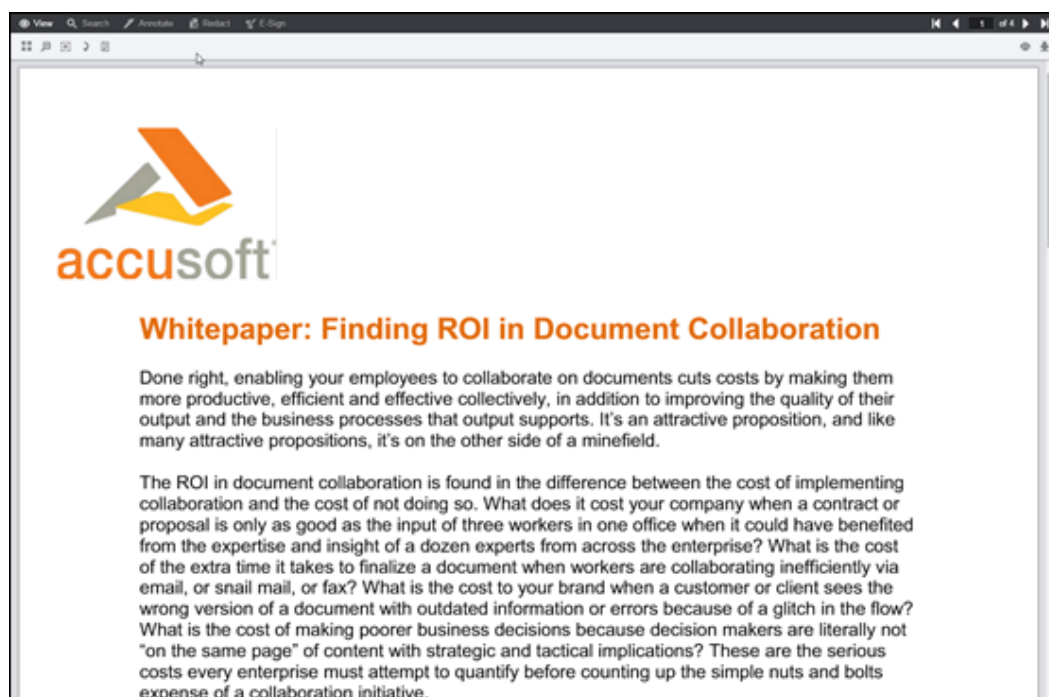
- o You can choose to load either the **Full Viewer**, the **Book Reader**, or the **Comparison Viewer**.

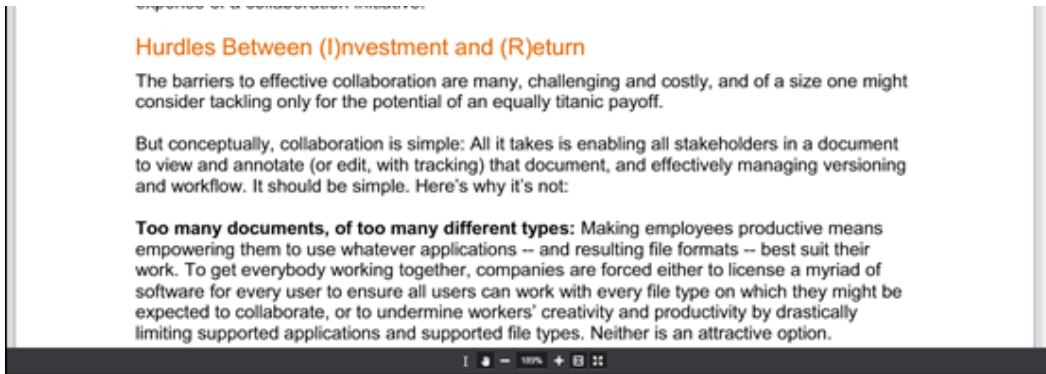
Select a sample document -OR- upload a document:

- o You can choose any of the 5 sample documents (Word, PDF, CAD, Tiff, or JPEG).
- o Or, you can upload a document from an arbitrary location on your computer. Note that dragging and dropping a file on this page is not supported in Internet Explorer 8.

2. Full Viewer

If you select Full Viewer on the splash page, then documents will be viewed with the full-featured, out-of-the-box responsive Viewer:





3. Book Reader

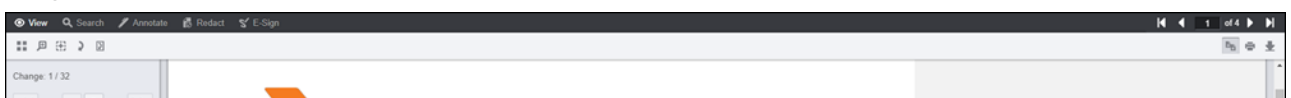
If you select **Book Reader** on the splash page, then documents will be viewed with the book reader. The Book Reader demonstrates how the Viewer can be heavily customized:

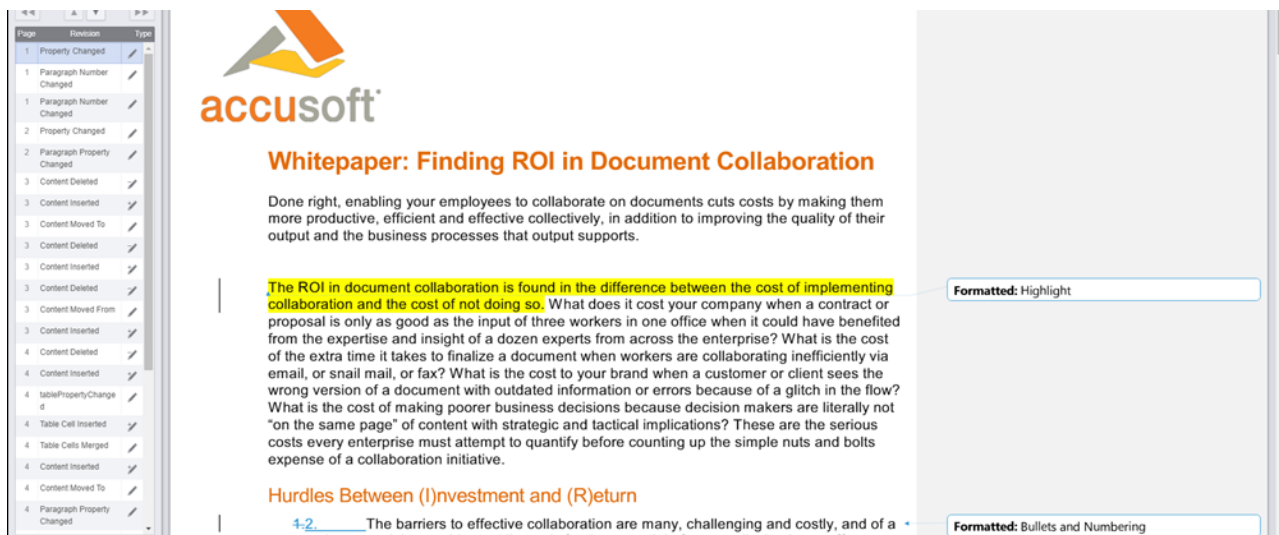


4. Comparison Viewer

NOTE: This feature requires a [Microsoft Office enabled PrizmDoc Viewer License](#).

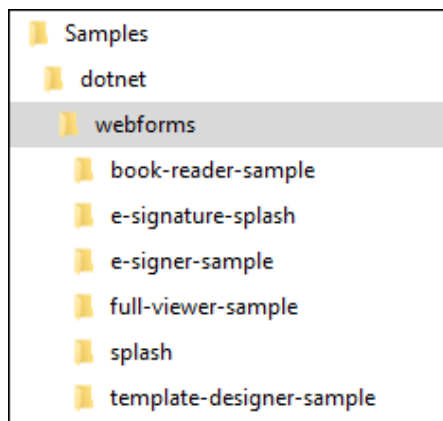
If you select **Comparison Viewer** on the splash page, then you can upload two Word (.doc or .docx) files to be compared in the Viewer. The Comparison Viewer shows how two Word documents can be reviewed and compared using PrizmDoc Viewer:





Directory Structure

The samples are installed at `C:\prizm\Samples\dotnet\webforms`. This folder contains 6 sub-folders, one folder for each of the four samples (full Viewer, book reader, e-signer and the e-signer template designer) and two folders for the splash pages (main splash page and the e-sign splash page):



Each of the sample folders are completely self-contained, meaning that they contain all of the files needed to run the sample. Furthermore, with the exception of a few project files and build files, the sample folders contain only the files needed to run the sample.

Folder contents: full-viewer-sample

NOTE: The full viewer (with the `comparisonMode` configuration parameter set to true) is used for document comparison.

File / Folder	Description
App_Code folder	Contains classes that support the communication between the Viewer and PAS. While the code for the classes can be modified as needed, modifications should be done with care. See <code>PrizmApplicationServices.cs</code> to see how we integrate the sample with PAS and see <code>PccConfig.cs</code> to see how we load the <code>pcc.config</code> configuration file. The files in this folder are essential and must be re-distributed to run the full Viewer.
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, and images that make up the Viewer.
viewer-assets/src folder	Contains Less, icons, languages, and HTML templates that can be used to build the Viewer CSS and customizations. This folder is non-essential, and does not need to be re-distributed.

File / Folder	Description
viewer-assets/Gulpfile.js	Contains Gulp tasks to build the Viewer Less, icons, and HTML templates. This file is non-essential and does not need to be re-distributed.
viewer-assets/package.json	A file used by npm (a package manager). It defines the dependencies installed by npm, which are required to run Gulp and compile the Viewer Less, icons, and HTML templates. This file is non-essential and does not need to be re-distributed.
viewer-webtier folder	Contains files that implement the ASP.NET layer of communication between the Viewer and PAS.
viewer-webtier/pcc.ashx	This file handles all incoming requests from the Viewer. This file simply uses the <code>App_Code/PrizmApplicationServices.cs</code> class to forward all requests to PAS.
viewer-webtier/pcc.config	Defines the connection settings for PAS.
Default.aspx, Default.aspx.cs	The default page for the sample. This page loads the full Viewer.
web.config	Contains IIS settings.
predefinedSearch.json	This data file contains information defining search queries that will appear as selectable items in the full Viewer. NOTE: This file is consumed by the page <code>Default.aspx</code> and the JSON is injected into the HTML that is returned by <code>Default.aspx</code> . Ultimately, the predefined search terms are provided as a JavaScript hash, when the Viewer is created.
redactionReason.json	This data file contains information defining redaction reasons that are available in the Viewer. NOTE: This file is consumed by the page <code>Default.aspx</code> and the JSON is injected into the HTML that is returned by <code>Default.aspx</code> . Ultimately, the redaction reasons are provided as a JavaScript hash, when the Viewer is created.)
Global.asax	The <code>Global.asax</code> file, also known as the ASP.NET application file, is a file that contains code for responding to application-level events raised by ASP.NET or by <code>HttpModules</code> . The <code>Global.asax</code> file resides in the root directory of an ASP.NET-based application. We use this file to initialize our <code>PccConfig</code> class.
full-viewer-sample.sln	Visual Studio solution file to open the sample.

Folder contents: book-reader-sample

File / Folder	Description
App_Code folder	Contains classes that support the communication between the Viewer and PAS. While the code for the classes can be modified as needed, modifications should be done with care. See <code>PrizmApplicationServices.cs</code> to see how we integrate the sample with PAS and see <code>PccConfig.cs</code> to see how we load the <code>pcc.config</code> configuration file. The files in this folder are essential and must be re-distributed to run the full Viewer.
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the book reader viewer. This file is non-essential and does not need to be re-distributed.
viewer-assets/less folder	Contains less that can be used to build the book reader CSS. This folder is non-essential, and does not need to be re-distributed.

File / Folder	Description
viewer-assets/Gruntfile	Contains Grunt tasks to build the reader LESS definitions. This file is non-essential and does not need to be re-distributed.
viewer-assets/package.json	A file used by npm (a package manager). It defines the dependencies installed by npm, which are required to run Grunt and compile the less.
viewer-assets/selection.json	A file used by the IcoMoon application to generate the icons in the book reader Viewer. If you need to add an icon to the Viewer, you can add the icon to this file and use the IcoMoon application (https://icomoon.io) to generate a new icon font. This file is non-essential and does not need to be re-distributed.
viewer-webtier folder	Contains files that implement the ASP.NET layer of communication between the Viewer and PAS.
viewer-webtier/pcc.ashx	This file handles all incoming requests from the Viewer. This file simply uses the <code>App_Code/PrizmApplicationServices.cs</code> class to forward all requests to PAS
viewer-webtier/pcc.config	Defines the connection settings for PAS.
index.html	The default page for the sample. This page calls the <code>pcc.ashx</code> handler to start a viewing session with PAS and then the page loads the Viewer.
sample-config.js	Contains references to the assets, web tier, and language files used by the Viewer in this sample.
web.config	Contains IIS settings.
Global.asax	The <code>Global.asax</code> file, also known as the ASP.NET application file, is a file that contains code for responding to application-level events raised by ASP.NET or by <code>HttpModules</code> . The <code>Global.asax</code> file resides in the root directory of an ASP.NET-based application. We use this file to initialize our <code>PccConfig</code> class.
book-reader-sample.sln	Visual Studio solution file to open the sample.

Folder contents: e-signer-sample

File / Folder	Description
App_Code folder	Contains classes that support the communication between the Viewer and PAS. While the code for the classes can be modified as needed, modifications should be done with care. See <code>PrizmApplicationServices.cs</code> to see how we integrate the sample with PAS and see <code>PccConfig.cs</code> to see how we load the <code>pcc.config</code> configuration file. The files in this folder are essential and must be re-distributed to run the full Viewer.
modules folder	Contains uncompiled assets of the Viewer. These files will be compiled to <code>viewer-assets/js/bundle.js</code> and <code>viewer-assets/css/bundle.css</code> by the build process defined in <code>Gulpfile.js</code> . The files in this folder are non-essential and do not need to be re-distributed.
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the Viewer.
Gulpfile.js	Contains Gulp tasks to build the viewer JS and CSS files. This file is non-essential and does not need to be re-distributed.
viewer-	A file used by npm (a package manager). It defines the dependencies installed by npm, which

File / Folder	Description
<code>assets/package.json</code>	are required to run Gulp and compile the viewer assets. This file is non-essential and does not need to be re-distributed.
<code>viewer-webtier</code> folder	Contains files that implement the ASP.NET layer of communication between the Viewer and PAS.
<code>viewer-webtier/pcc.ashx</code>	This file handles all incoming requests from the Viewer. This file simply uses the <code>App_Code/PrizmApplicationServices.cs</code> class to forward all requests to PAS
<code>viewer-webtier/pcc.config</code>	Defines the connection settings for PAS.
<code>index.html</code>	The default page for the sample. This page calls the <code>pcc.ashx</code> handler to start a viewing session with PAS and then the page loads the Viewer.
<code>web.config</code>	Contains IIS settings.
<code>webpack.config.js</code>	Webpack configuration file. This file contains all the settings for the webpack module bundler. We use webpack to compile all the files in the <code>modules</code> folder to the <code>bundle.js</code> and <code>bundle.css</code> that are found in the <code>viewer-assets</code> folder.
<code>Global.asax</code>	The <code>Global.asax</code> file, also known as the ASP.NET application file, is a file that contains code for responding to application-level events raised by ASP.NET or by <code>HttpModules</code> . The <code>Global.asax</code> file resides in the root directory of an ASP.NET-based application. We use this file to initialize our <code>PccConfig</code> class.
<code>e-signer-sample.sln</code>	Visual Studio solution file to open the sample.

Folder contents: `template-designer-sample`

File / Folder	Description
<code>App_Code</code> folder	Contains classes that support the communication between the Viewer and PAS. While the code for the classes can be modified as needed, modifications should be done with care. See <code>PrizmApplicationServices.cs</code> to see how we integrate the sample with PAS and see <code>PccConfig.cs</code> to see how we load the <code>pcc.config</code> configuration file. The files in this folder are essential and must be re-distributed to run the full Viewer.
<code>modules</code> folder	Contains uncompiled assets of the Viewer. These files will be compiled to <code>viewer-assets/js/bundle.js</code> and <code>viewer-assets/css/bundle.css</code> by the build process defined in <code>Gulpfile.js</code> . The files in this folder are non-essential and do not need to be re-distributed.
<code>viewer-assets</code> folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the Viewer.
<code>Gulpfile.js</code>	Contains Gulp tasks to build the viewer js and css files. This file is non-essential and does not need to be re-distributed.
<code>viewer-assets/package.json</code>	A file used by npm (a package manager). It defines the dependencies installed by npm, which are required to run Gulp and compile the Viewer assets. This file is non-essential and does not need to be re-distributed.
<code>viewer-webtier</code> folder	Contains files that implement the ASP.NET layer of communication between the Viewer and PAS.
<code>viewer-webtier/pcc.ashx</code>	This file handles all incoming requests from the Viewer. This file simply uses the <code>App_Code/PrizmApplicationServices.cs</code> class to forward all requests to PAS

File / Folder	Description
viewer-webtier/pcc.config	Defines the connection settings for PAS.
index.html	The default page for the sample. This page calls the <code>pcc.ashx</code> handler to start a viewing session with PAS and then the page loads the Viewer.
web.config	Contains IIS settings.
webpack.config.js	Webpack configuration file. This file contains all the settings for the webpack module bundler. We use webpack to compile all the files in the modules folder to the <code>bundle.js</code> and <code>bundle.css</code> that are found in the <code>viewer-assets</code> folder.
Global.asax	The <code>Global.asax</code> file, also known as the ASP.NET application file, is a file that contains code for responding to application-level events raised by ASP.NET or by <code>HttpModules</code> . The <code>Global.asax</code> file resides in the root directory of an ASP.NET-based application. We use this file to initialize our <code>PccConfig</code> class.
template-designer-sample.sln	Visual Studio solution file to open the sample.

Configuration with `pcc.config`

The file `pcc.config` is used to configure the connection settings between the web tier and PAS. The file can be found at: `<sample-folder-name>/viewer-webtier/pcc.config`. This file is self-documenting, but a little information about the configuration options is given below.

Option	Description
<code><DocumentPath></code> (Only in splash pages)	The sample pulls named documents from this location. The <code>DocumentPath</code> must have read/write permissions in order for the file drag and drop functionality of the splash page to work.
<code><PrizmApplicationServicesScheme></code>	Specifies the scheme (<code>http</code> or <code>https</code>) to use when connecting with PAS.
<code><PrizmApplicationServicesHost></code>	Specifies the host to use when connecting with PAS.
<code><PrizmApplicationServicesPort></code>	Specifies the port to use when connecting with PAS.

Development Information

The legacy ASP.NET WebForms sample has the following requirements for development:

- Visual Studio v2010 or later
- .NET 4.0 or later

Legacy JSP Sample

Legacy JSP Sample

Installation

This topic contains steps for how to install the legacy JSP sample on Linux and Windows.

NOTE: *JDK 1.7 and JRE 1.7+ are required.*

1. Run the appropriate "Client Installer" for your OS, [Windows](#) or [Linux](#).
2. Install [Apache Tomcat](#).
3. After installation is complete, launch **Tomcat Manager**.
4. In the WAR file to deploy section, select **Choose File**.
5. Select `PCCSample.war` file to upload from the installation location:
 - o For Windows: `C:\Prizm\Samples\jsp\target`
 - o For Linux: `/usr/share/prizm/Samples/jsp/target`
6. Click **Deploy**.
7. PrizmDoc Server web tier will be deployed on your Tomcat server. The deployed app may be found in the following location:
 - o For Windows: `C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps`
 - o For Linux: `/var/lib/tomcat7/webapps`
8. Give read/write permissions to the "Documents" folder, the "markup" folder and the "markupLayerRecords" folder. Give read permissions to the "imageStamp" folder. These folders are installed on the following location:
 - o For Windows: `C:\Prizm\Samples\`
 - o For Linux: `/usr/share/prizm/Samples`
9. Now you can browse to `http://localhost:8080/PCCSample` in your browser to see the JSP sample. See the [JSP Directory Structure](#) for more information on where these files are located.

Configure JSP Web Tier with SELinux

If you are running the JSP Web Tier on Security-Enhanced Linux you may experience issues when loading documents in the Viewer. The Web Tier needs to contact the PrizmDoc Server but SELinux disallows Tomcat processes from making outbound connections by default. You can run the following command to make the Tomcat domain permissive and allow the Web Tier to function normally with Tomcat:

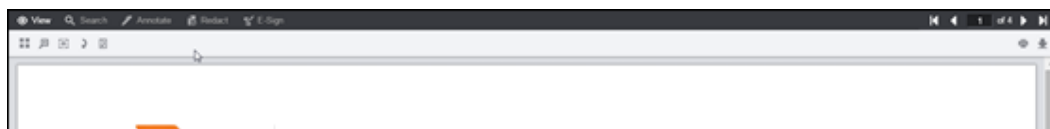
```
semanage permissive -a tomcat_t
```

Overview

From the splash page you have three options:

1. Choice of Viewer:
 - o You can choose to load either the **Full Viewer**, the **Book Reader**, or the **Comparison Viewer**.
Select a sample document -OR- upload a document:
 - o You can choose any of the 5 sample documents (Word, PDF, CAD, Tiff, or JPEG).
 - o Or, you can upload a document from an arbitrary location on your computer. Note that dragging and dropping a file on this page is not supported in Internet Explorer 8.
2. Full Viewer:

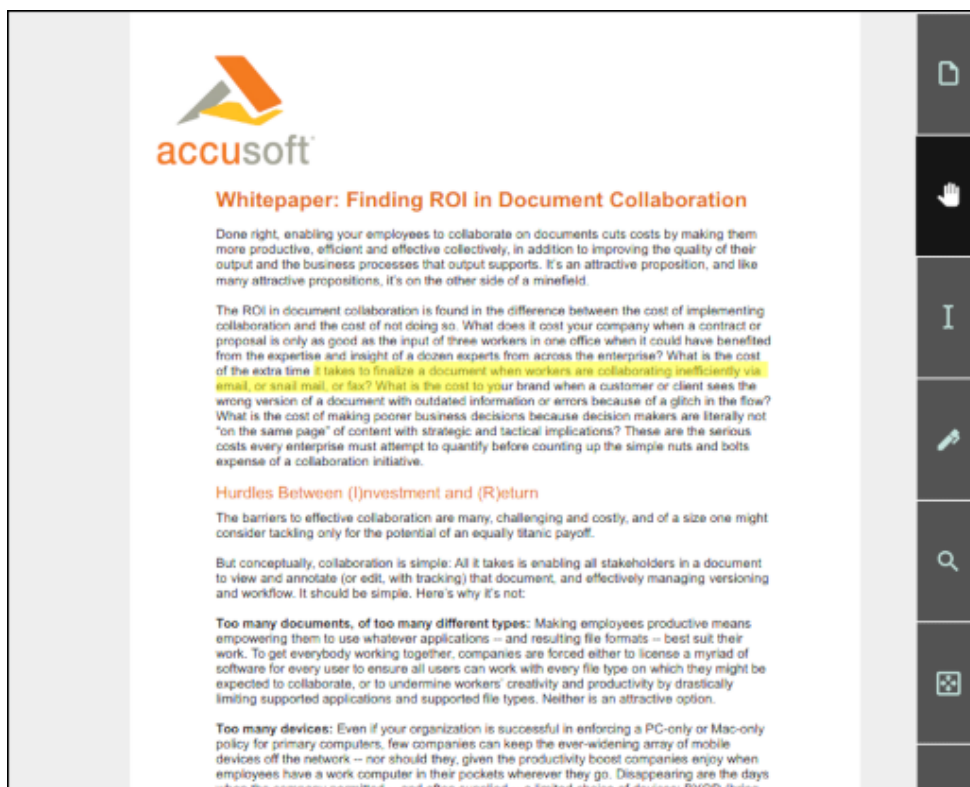
If you select **Full Viewer** on the splash page, then documents will be viewed with the full-featured, out-of-the-box responsive Viewer:

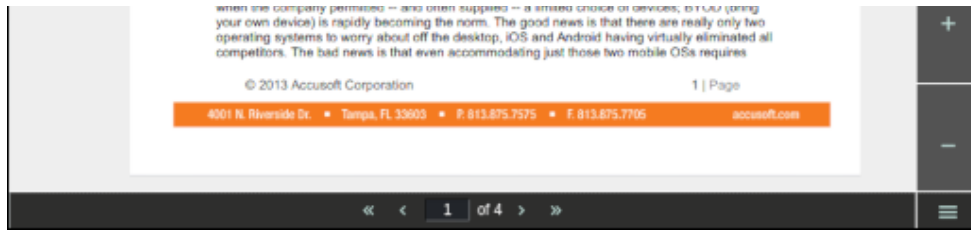




3. Book Reader:

If you select **Book Reader** on the splash page, then documents will be viewed with the book reader. The book reader demonstrates how the Viewer can be heavily customized:

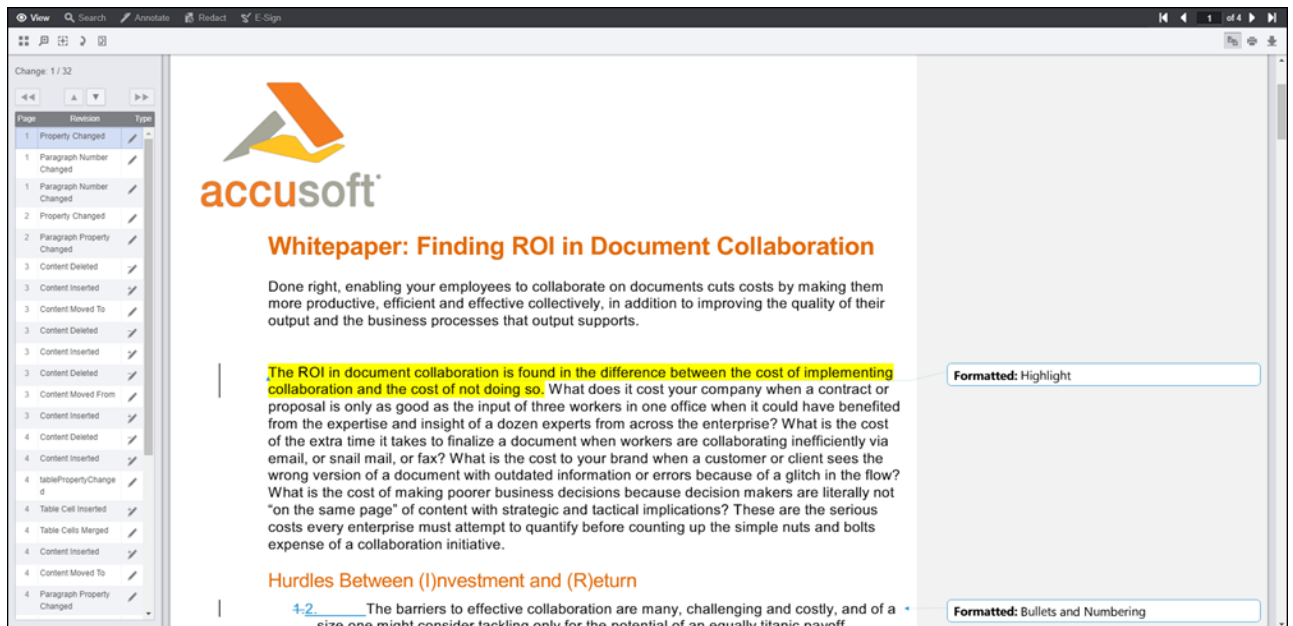




4. Comparison Viewer

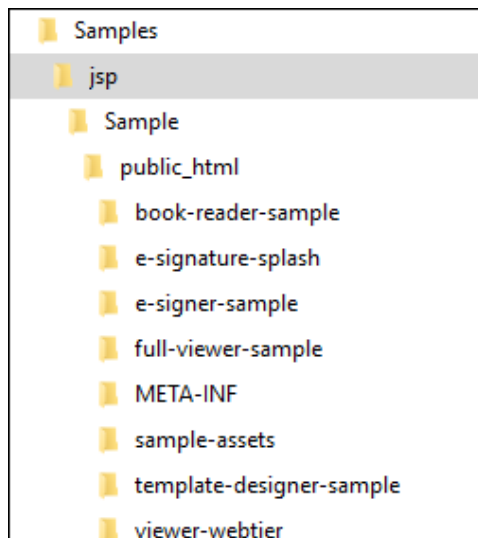
NOTE: This feature requires a [Microsoft Office](#) enabled PrizmDoc License.

If you select **Comparison Viewer** on the splash page, then you can upload two Word (.doc or .docx) files to be compared in the Viewer. The Comparison Viewer shows how two Word documents can be reviewed and compared using PrizmDoc:



JSP Directory Structure

The jsp samples are installed under `/usr/share/prizm/Samples/jsp/Sample/public_html`. This folder contains 6 sub-folders, one folder for each of the four samples (full Viewer, book reader, e-signer and e-signer template designer) and two folders for the splash pages (main splash page and the e-sign splash page):





Each of the sample folders are completely self-contained, meaning that they contain all of the files needed to run the sample. Furthermore, with the exception of a few project files and build files, the sample folders contain only the files needed to run the sample.

Folder contents: common

File / Folder	Description
src/com/accusoft/pccis/sample/pas and src/com/accusoft/pccis/sample/html5	Contains classes that support the communication between the Viewer and PrizmDoc Application Services. While the code for the classes can be modified as needed, modifications should be done with care.
public_html/WEB-INF/web.xml	Contains web tier settings. Some of these settings define the connection for the PrizmDoc Application Services. See more on the Configuration with web.xml section below.

Folder contents: full-viewer-sample

NOTE: The full viewer (with the `comparisonMode` configuration parameter set to `true`) is used for document comparison.

File / Folder	Description
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, and images that make up the Viewer.
viewer-assets/src folder	Contains Less, icons, languages, and HTML templates that can be used to build the Viewer CSS and customizations. This folder is non-essential, and does not need to be re-distributed.
viewer-assets/Gulpfile.js	Contains Gulp tasks to build the Viewer Less, icons, and HTML templates. This file is non-essential and does not need to be re-distributed.
viewer-assets/package.json	A file used by npm (a package manager). It defines the dependencies installed by npm, which are required to run Gulp and compile the Viewer Less, icons, and HTML templates. This file is non-essential and does not need to be re-distributed.
viewer-webtier folder	Contains files that implement the jsp layer of communication between the Viewer and the PrizmDoc Application Services.
viewer-webtier/pas.jsp	Handles all requests from the Viewer and forwards them to the PrizmDoc Application Services.
index.jsp	The default page for the sample. The Viewer's code gets loaded by this page.
predefinedSearch.json	This data file contains information defining search queries that will appear as selectable items in the full Viewer. NOTE: This file is consumed by the page <code>index.jsp</code> and the JSON is injected into the HTML that is returned by <code>index.jsp</code> . Ultimately, the predefined search terms are provided as a JavaScript hash, when the Viewer is created.
redactionReason.json	This data file contains information defining redaction reasons that are available in the Viewer. NOTE: This file is consumed by the page <code>index.jsp</code> and the JSON is injected into the HTML that is returned by <code>index.jsp</code> . Ultimately, the redaction reasons are provided as a JavaScript hash, when the Viewer is created.

Folder contents: book-reader-sample

File / Folder	Description
<code>viewer-assets</code> folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the book reader Viewer.
<code>viewer-assets/less</code> folder	Contains less that can be used to build the Viewer CSS. This folder is non-essential, and does not need to be re-distributed.
<code>viewer-assets/Gruntfile.js</code>	Contains Grunt tasks to build the viewer less. This file is non-essential and does not need to be re-distributed.
<code>viewer-assets/package.json</code>	A file used by npm (a package manager). It defines the dependencies installed by npm, which are required to run Grunt and compile the less.
<code>viewer-assets/selection.json</code>	A file used by the IcoMoon application to generate the icons in the book reader Viewer. If you need to add an icon to the Viewer, you can add the icon to this file and use the IcoMoon application (https://icomoon.io) to generate a new icon font. This file is non-essential and does not need to be re-distributed.
<code>viewer-webtier</code> folder	Contains files that implement the jsp layer of communication between the book reader Viewer and the PrizmDoc Application Services.
<code>viewer-webtier/pas.jsp</code>	Handles all requests from the Viewer and forwards them to the PrizmDoc Application Services.
<code>index.html</code>	The default page for the sample. The Viewer's code gets loaded by this page.
<code>sample-config.js</code>	Contains references to the assets, web tier, and language files used by the Viewer in this sample.

Folder contents: `e-signer-sample`

File / Folder	Description
<code>viewer-assets</code> folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the book reader Viewer.
<code>modules</code> folder	Contains uncompiled assets of the Viewer. These files will be compiled to <code>viewer-assets/js/bundle.js</code> and <code>viewer-assets/css/bundle.css</code> by the build process defined in <code>Gulpfile.js</code> . The files in this folder are non-essential and do not need to be re-distributed.
<code>Gulpfile.js</code>	Contains Gulp tasks to build the viewer js and css files. This file is non-essential and does not need to be re-distributed.
<code>viewer-assets/package.json</code>	A file used by npm (a package manager). It defines the dependencies installed by npm, which are required to run Gulp and compile the Viewer assets. This file is non-essential and does not need to be re-distributed.
<code>viewer-webtier</code> folder	Contains files that implement the jsp layer of communication between the book reader Viewer and the PrizmDoc Application Services.
<code>viewer-webtier/pas.jsp</code>	Handles all requests from the Viewer and forwards them to the PrizmDoc Application Services.
<code>index.html</code>	The default page for the sample. This page loads the Viewer.
<code>webpack.config.js</code>	Webpack configuration file. This file contains all the settings for the webpack module bundler. We use webpack to compile all the files in the <code>modules</code> folder to the <code>bundle.js</code> and <code>bundle.css</code> that are found in the <code>viewer-assets</code> folder.

Folder contents: `template-designer-sample`

File / Folder	Description
<code>viewer-assets</code> folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the book reader Viewer.
<code>modules</code> folder	Contains uncompiled assets of the Viewer. These files will be compiled to <code>viewer-assets/js/bundle.js</code> and <code>viewer-assets/css/bundle.css</code> by the build process defined in <code>Gulpfile.js</code> . The files in this folder are non-essential and do not need to be re-distributed.
<code>Gulpfile.js</code>	Contains Gulp tasks to build the viewer js and css files. This file is non-essential and does not need to be re-distributed.
<code>viewer-assets/package.json</code>	A file used by npm (a package manager). It defines the dependencies installed by npm, which are required to run Gulp and compile the Viewer assets. This file is non-essential and does not need to be re-distributed.
<code>viewer-webtier</code> folder	Contains files that implement the jsp layer of communication between the book reader Viewer and the PrizmDoc Application Services.
<code>viewer-webtier/pas.jsp</code>	Handles all requests from the Viewer and forwards them to the PrizmDoc Application Services.
<code>index.html</code>	The default page for the sample. This page loads the Viewer.
<code>webpack.config.js</code>	Webpack configuration file. This file contains all the settings for the webpack module bundler. We use webpack to compile all the files in the <code>modules</code> folder to the <code>bundle.js</code> and <code>bundle.css</code> that are found in the <code>viewer-assets</code> folder.

Configuration with `web.xml`

The file `web.xml` is used to configure the connection settings between the web tier and PrizmDoc Application Services. The file can be found at: `public_html/WEB-INF/web.xml`. This file is self-documenting, but a little information about the configuration options is given below.

Option	Description
<code>DocumentPath</code>	The sample pulls named documents from this location. The <code>DocumentPath</code> must have read/write permissions in order for the file drag and drop functionality of the splash page to work.
<code>PrizmApplicationServicesScheme</code>	Specifies the scheme (<code>http</code> or <code>https</code>) to use when connecting with PAS.
<code>PrizmApplicationServicesHost</code>	Specifies the host to use when connecting with PAS.
<code>PrizmApplicationServicesPort</code>	Specifies the port to use when connecting with PAS.

All settings are configured using the `<context-param/>` element. Here is an example of how it would look on the `web.xml` file:

Example

```
<context-param>
  <description>Prizm Application Services Scheme</description>
  <param-name>PrizmApplicationServicesScheme</param-name>
  <param-value>http</param-value>
</context-param>
```

Legacy Viewers

Introduction

If you are self-hosting your PrizmDoc Viewer backend, we have two legacy viewers which you may find useful (these legacy viewers assume you are self-hosting the backend and are not currently designed to be easily configured with our cloud-hosted backend options). These legacy viewers are included in the "Client Installer".

The legacy viewers each demonstrate a variety product functionality:

- [Book Reader Viewer](#)
- [E-Signature Viewers](#)

Customize the E-Signature Viewers

This section contains the following information:

- [Viewer Modular Design](#)
- [Configure the E-Signature Viewers](#)
- [Build the E-Signature Viewers](#)
- [Fill in Fields Programmatically](#)

Viewer Modular Design

Overview

The Template Designer and E-Signer viewers follow a modular design. This section provides information regarding the modular design and the principles we follow when implementing the Viewer modules.

Modules are generalized, single-feature, and reusable. Modules should implement their feature as generically as possible, making the least amount of assumptions about the external code consuming that module. If you find that you must use "and" to describe what the module does, you should probably be writing two modules.

Modules should also be instance-safe and viewer-safe. Each viewer should be able to use multiples of the same module without adverse effects, and multiple viewers should be able to initialize the same module without adverse effects. When writing a module, each module will need to initialize itself in such a way that allows multiple instances of that module to run at the same time. This means that things like global state variables are not allowed; any global variables need to be static.

- [Core Module](#)
- [JS-Only Modules](#)
- [UI Modules](#)
- [Initializing](#)
- [Destroying](#)
- [Components](#)

Core Module

Each viewer will need its own core module. The purpose of the core is to set up some common API, initialize modules, and provide a basic page structure and module containers:

- Core will set up both the StateStore and EventStore (available as `stateStore` and `eventStore` on the Viewer object). These APIs are expected by all modules and need to be created by the code initializing the modules.
- Core will also create the general page layout (think large containers), and will size and position all modules. It will need to provide parent containers as part of initializing UI modules.
- Core will also attach all exported members of common-core.js, including `parseIcons` and `parseComponents` functions, to the Viewer object. These are auxiliary functions that are expected by all modules.

JS-Only Modules

JS-only modules are encouraged when there is specific business logic to take care of. This includes, but is not limited to, client-server communication, controller modules that translate one thing to another, and modules that manage a state in the background. These modules should consist of a single JavaScript file, or a main JavaScript file that calls out to one or more sub-modules.

These modules can listen to and trigger any EventStore events, and can listen to any ViewerControl events. These modules must never listen to DOM events.

UI Modules

UI modules present a specific user interface, and should be entirely contained inside one container. If you find that you need two containers in your implementation, you should probably be writing two modules. These modules should always take up 100% of their specified container. It is up to the code creating the module to size and position it, and not up to the module itself.

These modules can listen to and trigger EventStore events. It can also listen to DOM events of elements that are located inside the module's container. They must never listen to DOM events for elements that are located outside of the module's container. Though it is not encouraged, it could be okay at times to listen to ViewerControl events, although such logic should most likely be extracted to a separate JS-only module.

UI modules should take care of updating their own UI, and staying up-to-date with any viewer logic (such as changes in relevant state values).

Initializing

Each module should take up to two parameters when initializing. The first will be an instance of a viewer (such as that defined in a "core" module). The second parameter is an optional `options` parameter, which provides extra settings and values to the module, such as the DOM element that the module should use as a parent container. Modules should register all events that they need in order to complete their tasks. Modules should not rely on external triggers that are not expressly defined as events.

Destroying

Modules should provide a mechanism to allow them to be destroyed. Typically, this will be a `destroy` method available as the module's API.

Any event that is registered during initialization or throughout the lifespan of the module must be removed during the destroy. This includes, but is not limited to, event store events, `ViewerControl` events, all DOM events, and events and functions that are temporarily registered to perform a transient task, such as animation frame optimizations or timeouts. In the case of the last example, the developer should never assume that a transient event was already disconnected, as a module could be destroyed before the transient task has completed.

Any resources created during initialization or throughout the lifespan of the module must be removed and cleaned

up during the destroy. This includes, but is not limited to, DOM elements, CSS classes used on the parent container element, any amounts of global data being stored, pending web requests or other asynchronous tasks, and functions that exist as part of events. The last is very important, as we need to avoid memory leaks due to data staying in scope indefinitely.

Components

Components are special types of modules that provide some widely reusable canned behavior for a single conceptual thing. Components will most likely be needed for, but not necessarily confined to, polyfilling native browser controls and components. Examples of this are the `TextInput`, `CheckboxCollection`, and `Dropdown` components. For example, browsers provide a native dropdown through the `select` tag; however, these are not all that pretty and have very limited styling and extensibility options. In order to provide flexible, extensible, and beautiful dropdowns, we have implemented the `Dropdown` component to polyfill the parts that a native `select` element does not provide.

When implementing components, we should provide an experience as close as possible to the native browser ability. For example, in dropdown, we need to provide a similar developer experience to using the native `select` tag. In this case, the developer using `Dropdowns` should provide a parent tag defining the component, as well as a list of elements to use as the options. Code to handle selecting options, including the label and dropdown arrow, as well as any extra markup, should be created by the component code.

Initializing Components

Components are initialized by providing the component parent element to the component. In the `Dropdown` example, this is the element equivalent to the `select` tag. In the case of sets, such as `CheckboxCollection` and `ButtonSet`, each element from the set is initialized separately, and it is up to the component to group them together in order to add functionality. Components should use the "Data Dash" DOM API (e.g., `data-pcc-something`) in order to define properties of that component.

Destroying Components

Similar to modules, components need to expose a `destroy` method which cleans up all resources used by that component.

Component API

Components should expose similar functionality to the native ability that they are polyfilling. This includes, but is not limited to, useful events (such as the "change" event for values), ways of getting and setting the value (in the case of input components), and a way to access the list of values. This should be standard throughout all components, as much as possible.

Configure the E-Signature Viewers

Introduction

You can configure the E-Signature viewers (Template Designer and E-Signer) with one of the following options. You can configure:

- [The Viewer parameters](#)
- [The Viewer control parameters](#)

NOTE: If you specify the `documentID` Viewer control parameter, it is still necessary to specify the `templateDocumentId` Viewer parameter.

Option 1 - Configure Options to Set When the Viewer is Built

Edit the `sample-config.js` file available in any of the legacy samples included with the "Client Installer".

Option 2 - Configure Options without Building the Viewer

Set `window.pccViewerConfig` before the Viewer is loaded. Note that the Viewer is loaded when all DOM elements are available (that is, when the jQuery document ready event fires). Any `window.pccViewerConfig` options you set will be used instead of the `sample-config.js` module settings (described in #1) or the query parameters (form or document).

For example, you could update `C:\Prizm\Samples\dotnet\mvc\viewers\template-designer-sample\index.html` to include the following JavaScript code to configure the following options in the C# Template Designer:

- Viewer control parameter for hiding side handles (instead of corner handles) when the handles are closed.
- Viewer control parameter for displaying the pages in a single horizontal row (instead of a vertical column).
- Viewer parameter for loading the document `PdfDemoSample.pdf` (instead of having to specify the document as a query parameter).

Example

```
<script type="text/javascript">
  window.pccViewerConfig = {
    markHandleMode: 'HideSideHandlesWhenClose',
    pageLayout: 'Horizontal',
    templateDocumentId: 'PdfDemoSample.pdf',
  };
</script>
```

Option 3 - Manually Embed the Viewer

You can disable default embedding of the Viewer and instead use your own code to embed the Viewer into a web page as follows:

1. Change the `id` property on the div reserved for embedding to something other than the default `pcc-viewer`; this will disable auto-embedding.
2. Create a separate JavaScript file that will hold all of the code for embedding. You can check `viewer-init.js` for an example. Make sure you reference the `id` that you specified in your html markup, as shown below:

Example

```
var viewer = $('#pcc-viewer-custom').pccESigner(options);
```

3. Reference your JavaScript file you created in Step #2 in your web page, after the `bundle.js` reference.

Example

```
<head>
  ...
  <!-- load the viewer bundles -->
  <link rel="stylesheet" href="viewer-assets/css/bundle.css">
  <script src="viewer-assets/js/bundle.js"></script>
  <script src="viewer-assets/js/your-javascript-file.js"></script>
  ...
</head>
<body>
  <div id="pcc-viewer-custom"></div>
</body>
```

Build the E-Signature Viewers

Introduction

The Template Designer Viewer and E-Signer Viewer work out-of-the-box, but if you want to customize either of these viewers, you will need to build them as follows:

1. Install **node.js**, which you can download from <https://nodejs.org/>.
2. Open a **node.js command prompt** and change to the directory of the Viewer you are building. For example, if you want to build the C# template designer, change to the C# template designer sample folder, as demonstrated below:

```
cd C:\Prizm\Samples\dotnet\mvc\viewers\template-designer-sample
```

NOTE: By default, the C# template designer sample is installed to **C:\Prizm\Samples\dotnet\mvc\viewers\template-designer-sample**.

3. Run the following command, which will install the dependencies for building the Viewer:

```
npm install
```

4. To build the Viewer, use one of the commands described below:
5. To create a single developer build, use the following command:

```
gulp build
```

6. To run a watch task, use the following command. This will automatically rebuild the Viewer when any files are modified. Note that you will need to keep the command window open while the watch task is running:

```
gulp
```

7. When creating a single build or running a watch task, you can create production builds by adding a `-p` flag to the end of the command, as demonstrated below. Production builds will output minified source code (both JavaScript and CSS) and will not generate sourcemaps:

```
gulp build -p
```

8. When creating a single developer build or running a watch task, you can get native system notifications when the build is complete by adding an `-n` flag to the end of the command, as demonstrated below:

```
gulp -n
```

The build process uses some standard open-sourced tools. To learn more about these resources and how to use them, refer to the following:

- Gulp - <http://gulpjs.com/>
- Webpack - <http://webpack.github.io/>
 - [Integrating Webpack in Gulp](#)
- Less - <http://lesscss.org/>

Fill in Fields Programmatically

Introduction

The `StateModified` event fires when fields are filled in and the `ModifyState` event can be used to update filled-in field values.

The example below demonstrates using a `StateModified` event handler to get the filled-in values of two fields and fire the `ModifyState` event to fill in a third field with the sum:

Example

```
viewer.eventStore.on('StateModified', function (ev, data) {
  if (data.state === 'FieldList') {
    var value1 = parseInt(data.stateValue.fieldList\[1\].value);
    var value2 = parseInt(data.stateValue.fieldList\[2\].value);
    data.stateValue.fieldList\[3\].value = (value1 + value2).toString();
    viewer.eventStore.trigger('ModifyState', {
      state: 'FieldList',
      stateValue: data.stateValue
    });
  }
});
```

Customize the Book Reader Viewer

Introduction

This topic covers how to install and customize the Book Reader Viewer.

Installing the Book Reader Viewer

Be sure to run the appropriate "Client Installer" for your OS, [Windows](#) or [Linux](#). Once complete, the Book Reader Viewer files can be found in the following location:

- For Windows: `C:\Prizm\Samples\jsp\Sample\public_html\book-reader-sample`
- For Linux: `/usr/share/prizm/Samples/jsp/Sample/public_html/book-reader-sample`

Working with the LESS preprocessor

The Book Reader Viewer uses [LESS](#) to pre-process the CSS for the Book Reader Viewer. In order to facilitate using this pre-processor in a development environment, the following files are included in the **book-reader-sample/viewer-assets** folder:

- Gruntfile.js
- package.json
- In order to use these files, you will need to install [Node.JS](#) in your development environment. Open your command line interface in the **book-reader-sample/viewer-assets** folder. Then, you can run the following commands from a command line or terminal:

Example

```
npm install -g grunt-cli
npm install
```

- Next, you can use this command to build the CSS files required for production:

Example

```
grunt buildprod
```

- You can also build development files, which will include extra source maps helpful in debugging CSS:

Example

```
grunt builddev
```

- Finally, while developing, you may choose to run the task in such a way that it will automatically run whenever any of the Less files change, as such:

Example

```
grunt dev
```


The Less preprocessor will generate the following file(s):

- `css/style.css` - contains the Viewer styles
- `css/style.css.map` - (optional) only present when using one of the dev options

Customizing the Styles

The styles should be loaded in the Book Reader Viewer in the following order:

1. `viewercontrol.css`
2. `style.css`

Namespace

The Book Reader Viewer uses the class `.pccv` in order to namespace the styles it uses. In order to override any selector used in the Book Reader Viewer, your selector must begin with the class `.pccv`:

Example

```
/* Set the navigation tab bar to dark red */
.pccv .pcc-nav-tabset,
.pccv .pcc-nav-tabset .pcc-tab-item,
.pccv .pcc-status-bar { background: #5b100d; }
```

Organization

All Less files are in the `less` folder. These individual files are split out based on functionality and are named in a self-explanatory way. For example, styles related to the search functionality are held in the `less/components/search.less` file.

Variables

There are many variables contained in `less/components/variables.less`, which control things like the color scheme and icon sizing. These variables can be modified in order to propagate changes throughout the Book Reader Viewer.

Grid System

The Viewer uses a basic grid system to assist with the UI layout. Through a series of rows and columns the layout can scale dynamically. Rows are used to create horizontal groups of columns. Columns are created by defining the number of twelve columns you will span. For example, three columns would use three divs with a class of `.pcc-col-4`:

Example

```
<div class="pcc-row">
  <div class="pcc-col-4">Left</div>
  <div class="pcc-col-4">Center</div>
  <div class="pcc-col-4">Right</div>
</div>
```

viewercontrol.css

This file contains the styles required for using ViewerControl. This file should not be changed, but rather, should have any necessary rules overridden by your own CSS. If choosing not to use any of our viewers, and instead embedding ViewerControl directly in a custom integration, this CSS file is still required.

PAS

Overview

PAS is a layer in front of PrizmDoc Server which is responsible for creating viewing sessions, saving and loading of annotations, and creating viewing packages (pre-converted content). Like your web application, PAS has privileged access to storage that you own (like a file system or database).

For an introduction to how PAS is part of the overall viewing architecture, see the [Architecture Overview](#) in our Getting Started guide.

For viewing functionality, your web application should only make REST API calls to PAS. PAS will then make calls to PrizmDoc Server on your behalf to ensure the conversion work is done. The only time your application should call PrizmDoc Server directly is when you need to perform non-viewing work, such as converting a file or burning annotations into a document.

We support the ability to do pre-conversion and long term caching of documents in a central location. To use this functionality, an external instance of a database will need to be configured to run with PAS.

PAS Clustering

PAS is designed to scale out well. In order to install and run PAS on multiple servers, you will need to consider the following:

- You will need a load balancer to accept incoming requests and route them to any instance of PAS. Since any request can be handled by any instance in the cluster, any on-the-shelf load balancing solution should do the job.
- All instances of PAS will need to be configured to point to the same PrizmDoc Server deployment (whether a single instance of PrizmDoc Server or the load balancer in front of a PrizmDoc Server cluster).
- All instances of PAS will need to be configured with the same storage solution, in order to have access to the same data. This includes using shared filesystem storage, such as a NAS, for all filesystem files, as well as a shared database when working with [Pre-Conversion Services](#).

For more information on installing PAS on a cluster, refer to the topic [Run PAS on Clusters](#).

This section contains the following "How To" information:

- [Set up Your Database for use with PAS](#)
- [Handle Specific Routes with PAS](#)
- [Compare Documents with PAS](#)
- [Pre-Convert Documents](#)
- [Pre-Populate Fields in the E-Signature Viewer](#)
- [Work with Viewing Packages](#)

For information on administering PAS, refer to [Administering PrizmDoc Viewer > PAS](#).

Set up Your Database for use with PAS

Introduction

This topic covers how to set up your database for use with PAS. For information on configuring PAS to communicate with your database, see the [PAS Configuration](#) section in the help file.

Creating the required database tables

Some databases, like Microsoft SQL Server, require that the tables be created before PAS can use them. This is a manual step.

The easiest way to create the tables is to run the scripts available as part of PAS:

On Windows:

```
cd C:\Prizm\pas\db
createtables.cmd
```

On Linux:

```
cd /usr/share/prizm/pas/db
./createtables.sh
```

This will create the required tables in the database that is configured through the PAS configuration file. If you need to change any of the configuration temporarily when running these scripts (such as using a different user that has the required privileges to create tables), you can specify any of the `database.*` properties from the PAS configuration file as command line flags, as such:

```
createtables.cmd --user=createTablesUser --password=Pa55w0rd
```

```
./createtables.sh --user=createTablesUser --password=Pa55w0rd
```

NOTE: The create tables script should be run when upgrading PAS as well. This will update the database schema without modifying any data already stored in the database.

Advanced use

For advanced database administrators, you may want to inspect and manually run the SQL scripts to create tables. You can use this by adding the `--export-scripts` flag to the above commands, as such (the commands below work on both Windows and Linux using the appropriate script):

```
./createtables.sh --export-scripts --filepath=/path/to/script.sql
```

When running this command, the configured database will not be changed, and the script will be saved to the output file specified. These SQL scripts can be found:

On Windows:

```
C:\Prizm\pas\db\mssql-scripts  
C:\Prizm\pas\db\mysql-scripts
```

On Linux:

```
/usr/share/prizm/pas/db/mssql-scripts  
/usr/share/prizm/pas/db/mysql-scripts
```

Handle Specific Routes with PAS

Introduction

This topic covers how to handle specific routes from PAS using custom logic. PAS is designed to handle all viewer requests in an appropriate way. However, there might be some cases where you need to add custom logic in place of those handlers. Here, we will look at how the request rerouting is done in general, as well as some examples where handling routes in a custom way might be desirable.

Rerouting any request to PAS

In the API Reference section, every route has a `Routes` key defined. This is the key that allows handling this route using outside logic. It can be used to instruct PAS to forward that particular route to any HTTP endpoint that it can access. This is done through the configuration file. Assuming the standard install location:

- On Windows: `C:\Prizm\pas\pcc.win.yml`
- On Linux: `/usr/share/prizm/pas/pcc.nix.yml`

To reroute any route, you will need to know that route's key. In these examples, we will use `RouteKey` as that key. If you want to simply handle a route, you can use the following config:

```
routeHandlers.RouteKey.url: "http://myserver/some/route"
```

You can also define any header you would like used when the request is proxied. This can be useful when wanting to secure a specific route from outside users while still allowing PAS to use it. This can be configured as such:

```
routeHandlers.RouteKey.url: "http://myserver/some/route"  
# define any header you would like  
routeHandlers.RouteKey.headers.x-my-custom-secret: "c2hoLCBJJ20gYSBzZW5yZXQ="   
routeHandlers.RouteKey.headers.x-use-any-name: "YW55IG5hbWUsIHJlYWxseQ=="
```

PAS will proxy any request to `RouteKey` to the url specified in the config. Since PAS routes will sometimes include important information, such as document ids, markup ids, etc., the entire route from PAS will be appended to the

URL, as such:

```
// route as handled by PAS
http://localhost:3000/RouteKey/u1234/sOmEiDvAlUe

// it will be forwarded here, as per the above config
http://myserver/some/route/RouteKey/u1234/sOmEiDvAlUe
```

If you would prefer to defer handling of that request back to PAS, you can return a 202 `Accepted` response to instruct Application Services to continue handling that request as it would normally. Any other response that is returned by the handler will be forwarded back to the client.

NOTE: Since this rerouting is based on an HTTP REST API, the example code that shows how to handle the requests in the following examples will be provided in pseudo-code. It is appropriate to implement this code in any language you feel comfortable using, as long as it can be exposed through an HTTP interface.

Example: How to handle markup file delete permissions

The routes to delete any markup files in PAS are open to all users, since PAS does not know about system users. However, you can let PAS know about user access. In the configuration, you will want to handle the following routes as such:

```
routeHandlers.DeleteMarkupLayer.url: "http://myserver/user/deletepermission"
routeHandlers.DeleteFormDefinition.url: "http://myserver/user/deletepermission"
```

You could potentially register this route handler on your server:

```
DELETE http://myserver/user/deletepermission/{markupType}/{markupId}
```

Using the following pseudo-code, you could determine whether the user is allowed to complete this delete action and instruct PAS on how to continue:

```
var markupType = parsedRouteParameters.markupType; // "FormDefinitions" or
"MarkupLayers"
var markupId = parsedRouteParameters.markupId; // a guid value

if (userCanDelete(markupType, markupId)) {
    sendResponseStatus(202); // Accepted
    // This will tell Application Services to continue with the delete
operation
} else {
    sendResponseStatus(403); // Forbidden
    // The 403 response will be returned to the client
}
```

Example: How to handle creating sessions

Sometimes, you may not be able to give PAS access to your documents directly. For example, when documents are

in a custom content management system and you may want to handle creating sessions on your own. In this example, we will look at handling the legacy `CreateSession` route.

The following would be an example of the configuration:

```
routeHandlers.LegacyCreateSession.url: "http://myserver/documents"
```

You would register a handler, as such, on your server:

```
GET http://myserver/documents/CreateSession
```

You would be able to use the following pseudo-code to handle this request:

```
var document = parsedQueryParameters.document;
// also make sure to handle the case for the "form" query
parsedQueryParameters
// when using the e-sign viewers

var documentData = getDocumentData(document);

// use the ViewingSession API to create the actual session
var sessionResponse = makeRequest({
  method: "POST",
  url: "http://localhost:3000/ViewingSession"
  body: {
    source: {
      type: "upload",
      displayName: documentData.displayName,
      markupId: documentData.uniqueIdentifier
    }
  }
});

// optionally do this later or in a separate thread, for better performance
var uploadResponse = makeRequest({
  method: "PUT",
  url: "http://localhost:3000/ViewingSession/" +
sessionResponse.viewingSessionId + "/SourceFile",
  headers: {
    "Accusoft-Secret": "mysecretkey"
  },
  body: documentData.theDocument
});

// return the response back to PAS, to be forwarded to the client
returnResponse(sessionResponse);
```

Compare Documents with PAS

Performing Document Comparison

You can use PrizmDoc Viewer to show an end user a comparison view of two different Microsoft Office documents (if you are hosting the backend yourself, you must be running PrizmDoc Server on a Windows machine with a [Microsoft Office enabled PrizmDoc license](#)).

Here is how you typically set this up:

Step 1: Create a Comparison Viewing Session

As with normal viewing sessions, your web application begins by sending a POST `/ViewingSession` request to create a new viewing session. However, this time you will set the `source.type` to `"comparison"`. And, instead of providing only one document, you will provide two.

There are different ways to provide the two documents (refer to the information about document comparison in the [Viewing Sessions REST API documentation for PAS](#)). The most common and recommended way is to have your web application upload each of them in subsequent requests.

Here is how you would create a comparison viewing session and indicate that you intend to upload both documents in subsequent requests:

```
POST pas_base_url/ViewingSession
Content-Type: application/json

{
  "source": {
    "type": "comparison",
    "displayName": "Example Comparison",
    "original": {
      "type": "upload",
      "displayName": "original.docx"
    },
    "revised": {
      "type": "upload",
      "displayName": "revised.docx"
    }
  }
}
```

PAS will respond immediately, giving you a new `viewingSessionId`:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "viewingSessionId": "XYZ..."
}
```

Step 2: Initialize the Viewer

Now that you have the `viewingSessionId`, your web application can go ahead and render the HTML page with the viewer configured to use that `viewingSessionId`. The document content will load for the end user as soon as it becomes available.

But, in order for content to be shown, you still need to upload the two documents.

Step 3: Upload the Two Documents

When uploading the two documents, our API uses the terms *original* and *revised* to refer to the two documents. You issue two PUT requests: one to upload the *original* file, and one to upload the *revised* file:

```
PUT pas_base_url/v2/viewingSessions/XYZ.../sourceFile/original
<<file 1 bytes>>
```

```
HTTP/1.1 200 OK
```

```
PUT pas_base_url/v2/viewingSessions/XYZ.../sourceFile/revised
<<file 2 bytes>>
```

```
HTTP/1.1 200 OK
```

Once both documents have been uploaded, a background process will begin automatically creating a new document which is a visual comparison of the two. Once that comparison document is ready, it will internally be set as the actual, primary document of the viewing session, and the viewer running in the browser will load the first page of this automatically-created comparison document as soon as it becomes ready.

Additional Info

There are additional API endpoints which allow you to download the *original* and *revised* documents you initially provided, as well as an endpoint which our viewer uses to download metadata about the revisions in the document so the end user can navigate between them. Your application may be interested in using these endpoints directly. For more information, see the relevant PrizmDoc Server REST API documentation (these endpoints can be called via PAS, which proxies the requests to PrizmDoc Server):

- [GET /v2/viewingSessions/{viewingSessionId}/sourceFile/original](#)
- [GET /v2/viewingSessions/{viewingSessionId}/sourceFile/revised](#)
- [GET /v2/viewingSessions/{viewingSessionId}/revisionData](#)

Pre-Convert Documents

Pre-converting documents in PrizmDoc Application Services (PAS)

When viewing large documents, a user can experience a delay viewing later pages in the document. The Pre-conversion API allows the user to avoid any delay in viewing a fully converted document prior to the creation of a viewing session.

This section describes a typical use in pre-converting and management of the pre-converted Viewing Packages:

1. How to Create a Viewing Package by Pre-converting Documents.
2. How to Obtain Information for a Viewing Package.
3. How to Delete a Viewing Package

How to Create a Viewing Package by Pre-converting Documents

Pre-conversion is available by using the Pre-conversion API. For detailed information, refer to the [PrizmDoc Application Services RESTful Viewing Package Creators API](#) section.

Documents are pre-converted using the following steps:

Step 1

Issue a POST request with the body of the request containing JSON formatted 'source' object. The source.type property can be a "document", "url" or "upload".

In this example, "document" is used as a source.type property.

```
POST http://localhost:3000/v2/viewingPackageCreators
```

viewingPackageCreator POST Body

```
Content-Type: application/json
{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.doc",
      "documentId": "unT67FxeKm81k1p0kPnyg8",
      . . .
    },
    "viewingPackageLifetime": 2592000
  }
}
```

A successful response to the above POST provides a `processId` in the response body:

```
200 OK
Content-Type: application/json
{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.doc",
      "documentId": "unT67FxeKm81k1p0kPnyg8",
      . . .
    },
    "viewingPackageLifetime": 2592000
  },
  "expirationDateTime": "2015-12-09T06:22:18.624Z",
  "processId": "khjyrfKLj2g6gv8fdqg710",
  "state": "processing",
  "percentComplete": 0
}
```

Step 2

Using the `processId` obtained in the step 1, query the pre-conversion process for the status:

Example

```
GET http://localhost:3000/v2/viewingPackageCreators/khjyrfKLj2g6gv8fdqg710
```

A successful response body contains the JSON formatted properties `state` and `percentComplete`. The `state` value indicates whether it is `complete` or `processing` and the property `percentComplete` indicates percentage amount complete.

Start polling the status by issuing a GET command using the above URL. It is recommended to use shorter intervals initially between the requests for the first few times. If it is still not complete, then the document may be large, requiring more processing time.

In scenarios like this, an increase in the time interval between requests would be necessary to prevent a large number of status requests that could potentially cause network congestion. On 100% completion, the response body will among other information contain an output object with `packageExpirationDateTime` property:

```
200 OK
Content-Type: application/json
{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.doc",
      "documentId": "unT67FxeKm81k1p0kPnyg8",
      . . .
    },
    "viewingPackageLifetime": 2592000
  },
  "output": {
    "packageExpirationDateTime": "2016-1-09T06:22:18.624Z"
  },
  "expirationDateTime": "2015-12-09T06:22:18.624Z",
  "processId": "khjyrfKLj2g6gv8fdqg710",
  "state": "complete",
  "percentComplete": 100
}
```

How to obtain information about the converted Viewing Package

For obtaining detailed information about the converted Viewing Package, refer to the [PrizmDoc Application Services RESTful Viewing Packages API](#) section.

When the status is 100% complete, details can be obtained about the converted package by issuing the following request:

```
GET http://localhost:3000/v2/viewingPackages/unT67FxeKm81k1p0kPnyg8
```

Example Response Body

```
200 OK
Content-Type: application/json
{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.doc",
      "documentId": "unT67FxeKm81k1p0kPnyg8",
      . . .
    },
    "viewingPackageLifetime": 2592000
  },
  "state": "complete",
  "packageExpirationDateTime": "2016-1-09T06:22:18.624Z"
}
```

How to Delete a Previously Converted Package

For obtaining detailed information about deleting converted Viewing Package, refer to the [PrizmDoc Application Services RESTful Viewing Packages API](#) section.

A previously converted package can be deleted by issuing a DELETE request:

```
DELETE http://localhost:3000/v2/viewingPackages/unT67FxeKm81k1p0kPnyg8
```

This request marks the package for asynchronous deletion. A successful response is as follows:

```
204 (No Content)
```

Pre-Populate Fields in the E-Signature Viewer

Introduction

This example demonstrates automatically pre-populating pre-existing user data into form fields at runtime. Route the form definition GET request to a custom handler instead of Prizm Application Services (PAS) directly. In this handler, fetch the Form Definition from PAS, insert default values into the given form definition using external data, and return the modified form definition to the viewer. This example assumes PAS is listening on `localhost:3000`, but this will vary based on your server configuration.

Step 1: Load the Form Definition

Example

```
GET http://localhost:3000/FormDefinitions/5418c96283bc469783bd30e7c8fdc059
Content-Type: application/json
{
  "templateDocumentId": "Form 3.pdf",
```

```
"globalSettings": { ... global settings ... },
"formRoles": { ... form roles ... },
"groups": {},
"formName": "Form 1 - updated",
"formData": [ ... form data ... ]
}
```

Step 2: Get User Data

This will vary based on your data source. You might load data from a database, a file, or another location. Or, your GET request to load the form definition may have included a session ID for a particular user from which secondary information can be queried. Let's assume we receive the following user data object:

Example

```
{
  "Name": "John Smith",
  "Address": "123 Town St",
  "Phone": "(555) 555-5555"
}
```

Step 3: Insert the Data

We can iterate through each field in the `formDefinition`, check if there is data corresponding to that field ID in the example user data object, and set its `defaultValue` property appropriately depending on the field template type:

Example

```
//userData is the result of our example external data GET request
//formDefinition is the result of our PrizmDoc FormData GET request
formDefinition.formData = formDefinition.formData.map(function(field) {
  if (userData[field.fieldId]) {
    switch (field.template) {
      // Checkboxes are either "checked" or not
      case 'CheckboxTemplate':
        return extend({}, field, {
          defaultValue: userData[field.fieldId] ? 'checked' : ''
        });
      // Signatures use a different value based on their type,
      // but we will assume text for this example
      case 'SignatureTemplate':
      case 'InitialsTemplate':
        return extend({}, field, {
          defaultValue: {
            type: 'text',
            value: userData[field.fieldId],
            fontName: 'Grand Hotel'
          }
        });
      // Date templates use an ISO datetime
      case 'DateTemplate':
```

```
        return extend({}, field, {
            defaultValue: (new Date(userData[field.fieldId])).toISOString()
        });
        // Text templates use a string
        case 'TextTemplate':
            return extend({}, field, {
                defaultValue: userData[field.fieldId]
            });
        }
    }
    // If the item was not in the database, return it as-is.
    return field;
});
```

NOTE: A field collection has the option to "Allow Multiple Selections". If "Allow Multiple Selections" is false, but multiple fields in that group are set to pre-populate as "checked", the first field with a `defaultValue` of "checked" will be set as the only selected field for that collection.

Step 4: Return Data to the Caller

Return the updated `formDefinition` object to the function, web service, or other source that called it. When the data reaches the viewer and the `FormLoaded` event in the E-Signature Viewer fires, fields in this form definition with a valid `defaultValue` will be populated.

Work with Viewing Packages

Introduction

A Viewing Package is a cached version of a document that the Viewer will use when viewing a document. Viewing a document from a Viewing Package significantly reduces the load on PrizmDoc Server and allows you to serve many more users per minute than you could otherwise. A Viewing Package can be created through [Pre-Conversion](#) or by using [On-Demand Caching](#).

This topic provides information about the following:

- [Storage](#)
- [Configuration](#)
- [Pre-Conversion](#)
- [On-Demand Caching](#)
- [Performance Considerations](#)
- [Known Issues](#)

Storage

Viewing Packages are stored in both the filesystem and configured database. If using multiple instances of PAS, you must use a shared database and NAS (Network Attached Storage). The [Running PrizmDoc Application Services \(PAS\) on Multiple Servers](#) topic can provide more information for configuring PAS in Cluster Mode.

By default, storage is configured in the following way:

Config Key	Storage Provider	Description
viewingPackagesData	database	Data about a Viewing Package. This is the data that can be retrieved from GET /v2/viewingPackages.
viewingPackagesProcesses	database	Data about a Viewing Package creator process. This is the data that can be retrieved from GET /v2/viewingPackageCreators.
viewingSessionsData	database	Data about a Viewing Session. When creating a Viewing Session, an entry is added to this table.
viewingSessionsProcessesMetadata	database	Data about processes for a Viewing Session. This is currently used for content conversion and markup burner processes.
viewingPackagesArtifactsMetadata	database	Metadata for a viewing package artifact. This is used to find specific artifacts for a package and contains the artifact type, the file name in the filesystem among other important information.
viewingPackagesArtifacts	filesystem	Artifacts for a Viewing Package. These include SVG and raster content for every page, the source document, and other artifacts the Viewer will likely request.

Configuration

Viewing Packages are opt-in and require special configuration to work properly. At a minimum, your configuration should include the following:

```
\# Feature toggles
feature.viewingPackages: "enabled"

\# Database configuration
database.adapter: "sqlserver"
database.host: "localhost"
database.port: 1433
database.user: "pasuser"
database.password: "password"
database.database: "PAS"

\# Default timeout for the duration of a viewing session
defaults.viewingSessionTimeout: "20m"

viewingPackagesData.storage: "database"
viewingPackagesProcesses.storage: "database"
viewingSessionsData.storage: "database"
viewingSessionsProcessesMetadata.storage: "database"
```

```
viewingPackagesArtifactsMetadata.storage: "database"
viewingPackagesArtifacts.storage: "filesystem"
viewingPackagesArtifacts.path: "/your/path/to/viewingPackages"
```

Pre-Conversion

Creating a Viewing Package through Pre-Conversion provides a way to generate packages whenever it makes the most sense for your application to do so. It allows you to make use of down-time for Pre-Conversion to reduce load in high traffic periods. Pre-Conversion does all the work of creating a Viewing Package whenever it is requested. It starts a process that will begin downloading content and allow you to poll for progress.

It is recommended that you maintain a queue of Pre-Conversions so you don't overload the server and have faster turnaround time. We recommend a maximum of 5 Pre-Conversion processes at a time per PAS and PrizmDoc Server instance. This will allow packages to be created quickly while maintaining a sustainable load.

For a step-by-step process, go to the [Pre-Converting Documents](#) topic.

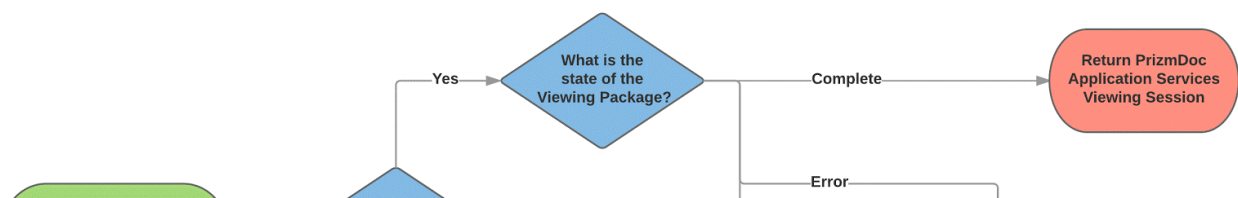
On-Demand Caching

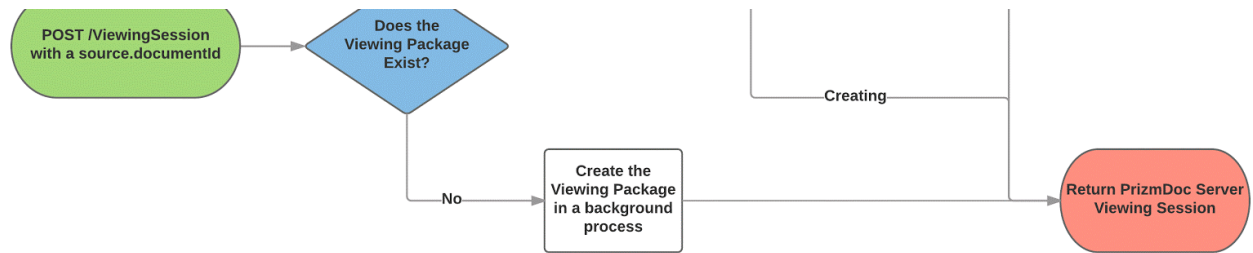
Creating a Viewing Package through On-Demand Caching is a seamless process through the [Viewing Session API](#). On-Demand Caching allows you to trigger a Viewing Package creation process in the background and use the resulting Viewing Package when it is ready. This feature is designed to allow immediate viewing using PrizmDoc Server while caching a package for subsequent views of the same document.

As an example, consider this request for a Viewing Session:

```
POST /ViewingSession
{
  "source": {
    "documentId": "PdfDemoSample-alb0x19n2",
    "type": "document",
    "fileName": "PdfDemoSample.pdf"
  }
}
```

This request will always return a `viewingSessionId` regardless of the status of the matching Viewing Package. If a Viewing Package does not currently exist with the given `documentId`, PrizmDoc Server will handle document viewing while a background process creates a Viewing Package. Once the background process is complete, PAS will handle all further viewing sessions until the Viewing Package expires (24 hours by default).





Performance Considerations

When enabling the Viewing Packages feature, there are additional considerations for hardware requirements. See [Sizing Servers](#) for more information.

Known Issues

Document Comparison

Viewing packages are not supported for comparison viewing sessions.

Markup & Redaction

When creating a Viewing Package and pages of the source document have successfully converted to the intermediate PDF, but PAS fails to get one or more pages of viewable content, the Viewer will not have that failed page content available to markup (i.e., redact). Because of this, when the user attempts to burn the markup into the output document, (which is generated from the intermediate PDF), it will have the page content and no markup.

If this occurs and the failed page contains content that must be redacted, a new viewing package should be created to retry the conversion.

Raster Content

- Viewing Packages limit available raster content. During Viewing Package creation, raster content is requested only for 0.125 , 0.25, 0.5, and 1.0 scale. API requests for raster content with a scale that is not one of these values will return 404 Not Found. The Viewer will always request full scale raster content from a viewing package which may result in degraded fidelity since the full scale image will be returned and any additional scaling will occur within the browser. Additionally, requests for [tiling](#) and [thumbnails](#) are not supported and will return 501 NotImplemented.
- The [render.html5.alwaysUseRaster](#) property cannot be used when creating a viewing session with caching.

Watermarks

- The watermarks property cannot be used when creating a viewing session with caching.

PrizmDoc Server

This section contains the following information:

- [Use the PrizmDoc Server API](#)
- [Markup Burner XML Specification](#)
- [Markup JSON Specification](#)
- [How To Examples](#)
 - [Compare Documents](#)
 - [Convert Content with Content Conversion Service](#)
 - [Content Conversion Demo](#)
 - [How to Configure the Demo on Windows](#)

- [How to Configure the Demo on Linux](#)
- [Migrate from PrizmDoc Cloud Servers to PrizmDoc Viewer Self-Hosted Servers](#)
- [Perform Auto-Redaction](#)
- [Set up a Viewing Session for a CAD Drawing which has XREF Dependencies](#)
- [Use the Markup JSON Schema](#)
- [Use a Viewing Session](#)
- [Watermark Content in a Viewing Session](#)

Use the PrizmDoc Server API

Introduction

PrizmDoc Server exposes a [RESTful API](#) that clients may use to perform operations on documents, enabling them to manipulate and extract content. In the same way that the PrizmDoc Server API is used by the Viewer, Developers can use the API to create their own applications to generate viewable content, perform redactions, and convert documents.

Resources

The PrizmDoc Server API exposes several resources that are designed to work together to perform document operations. This overview of these resources will provide developers with the understanding required to quickly get up and running with PrizmDoc Services:

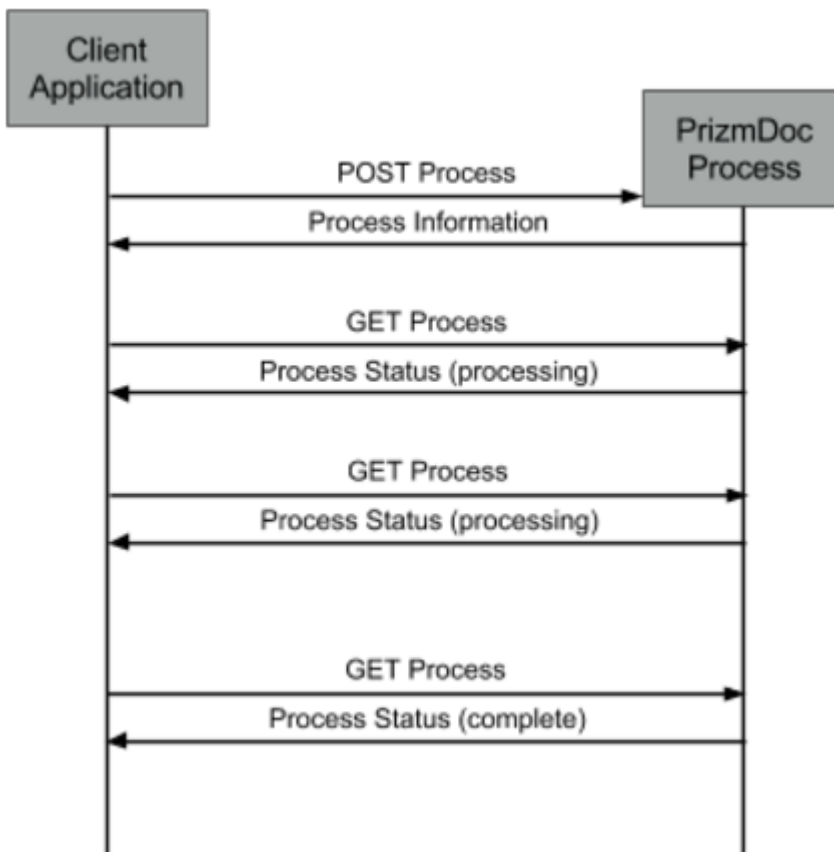
Work Files

[Work Files](#) form the basis of the PrizmDoc Service temporary document storage. Documents provided by users, converted content results, redaction markup, and burned output are all stored within Work Files. Documents are stored and retrieved through requests to the Work File Service made by user applications or other PrizmDoc Services. Work Files are not intended for long-term storage and are periodically removed when they are no longer required for PrizmDoc Server operations.



Processes

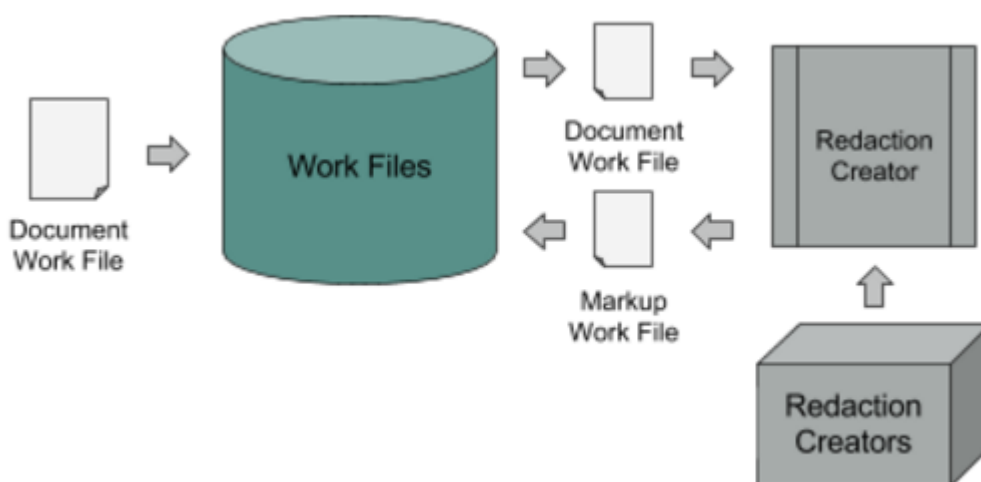
Several PrizmDoc Server operations are represented as asynchronous work processes. These processes are created by a request containing configuration information and immediately begin their work. Prior to completion, information about the process can be requested in order to determine its current state. Processes eventually reach a terminal state, at which point either the process was successful and the output is available or the process failed and is in an error state. After completion, processes do remain available for a short time, but since their work is done, they are soon removed by periodic maintenance. The lifetime of a process is configurable within its creation options.



These processes include:

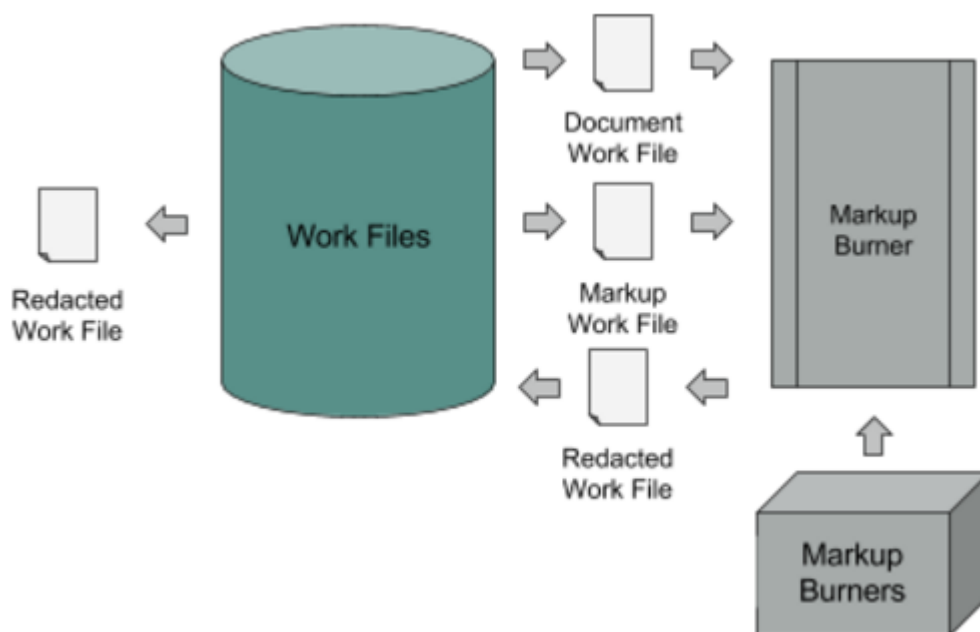
Redaction Creators

[Redaction Creators](#) are processes that operate on a document Work File to generate a document markup Work File. The user provides regular expressions that identify the text to be redacted, and the process generates a markup Work File that contains data describing the locations of the matches. The generated markup Work File can then be used in a Markup Burner process to create a document with the redactions burned in.



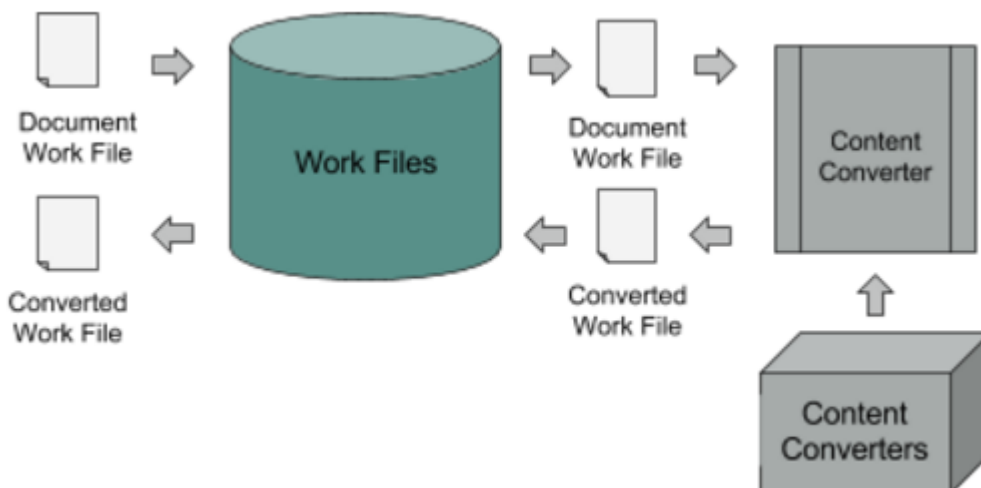
Markup Burners

[Markup Burners](#) are processes that create a "burned" document Work File as output from a source document Work File and a markup Work File describing redactions or annotations.



Content Converters

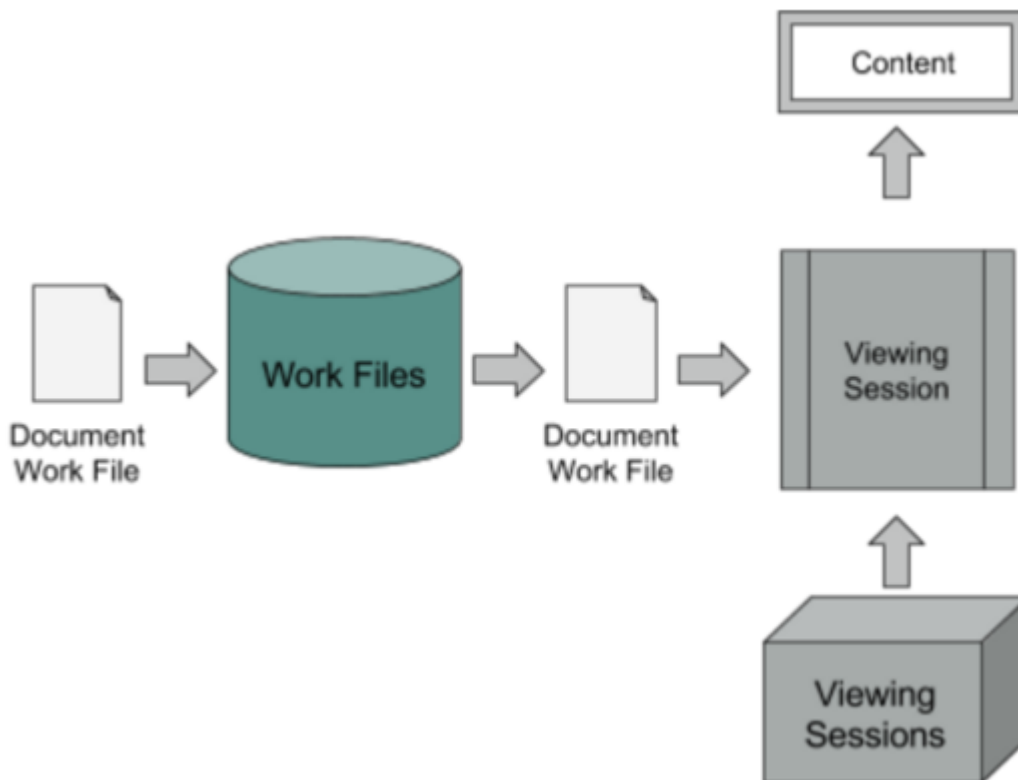
[Content Converters](#) are processes that convert one or more source Work File documents to one or more output Work File documents. Many different document formats are supported as source documents, and several common formats are allowed for output. After process completion, converted output is available as one or more Work Files.



Viewing Sessions

[Viewing Sessions](#) create converted content primarily for, but certainly not restricted to, use by the Viewer. They provide access to [SVG and raster renderings of document pages as well as any extracted document text, metadata for the converted content](#), and [attachments](#) in the case of email documents. Viewing Sessions are typically short-lived (20 minutes by default), but their lifetimes are configurable (see the `viewing.sessionLifetime` property in [Central Configuration](#) for details). Viewing Sessions also provide access to other document processes not directly

related to viewing.



PrizmDoc Server Operations

The PrizmDoc API also provides server level operations that allow the user to query the [health status](#) of sub-services as well as the overall system health. When using several PrizmDoc Server instances in a [cluster environment](#), APIs exist to query and configure the servers that each instance can use to route traffic for load balancing.

Markup Burner XML Specification

Introduction

A user can provide a specification of required modifications to a document in an XML-based format. The following nodes are expected as part of the declaration:

```
<?xml version="1.0"?>
<documentAnnotations>
  <pages>
    <page id="1" pageWidth="612" pageHeight="792"></page>
  </pages>
</documentAnnotations>
```

A `<pages>` element can contain a number of `<page>` elements, each containing an arbitrary number of the `<annotation>` elements. Each `<page>` element must have the following attributes:

Name	Type	Required	Description
<code>id</code>	integer	yes	A 1-based page number; annotations that are listed as child elements of a given <code><page></code> are only applied to the corresponding page of the target document.
<code>pageWidth</code>	floating point	yes	Page width. This attribute is only used for positioning of annotations which use absolute coordinates (such as <code><line-rectangles></code> data in text_hyperlink_annotation). This value can be arbitrary, as long as absolute coordinates specified in annotations are consistent with it.
<code>pageHeight</code>	floating point	yes	Page height. This attribute is only used for positioning of annotations which use absolute coordinates (such as <code><line-rectangles></code> data in text_hyperlink_annotation). This value can be arbitrary, as long as absolute coordinates specified in annotations are consistent with it.

Each `<page>` can contain an arbitrary number of the `<annotation>` elements.

Each `<annotation>` element defines a certain modification to a page content. A specific set of attributes depends on `drawType` attribute of the `<annotation>`.

Common Attributes

Each `<annotation>` element supports the following attribute set:

Name	Type	Required / Default	Description
<code>drawType</code>	string	yes	One of the supported draw types (see Draw Type Descriptions below) that defines what modification should an annotation apply to a page.
<code>x_percent</code>	float	0.0	X position of a bounding rectangle, in 0-to-1 based portion of a page width.
<code>y_percent</code>	float	0.0	Y position of a rectangle, in 0-to-1 based portion of a page height.
<code>width_percent</code>	float	0.0	Width of a bounding rectangle, in 0-to-1 based portion of a page width.
<code>height_percent</code>	float	0.0	Height position of a rectangle, in 0-to-1 based portion of a page height.

The set of `x_percent`, `y_percent`, `width_percent`, `height_percent` coordinates is referred to as a "bounding rectangle" in the [draw type descriptions](#) below.

Example XML

Below is a full working sample of a redaction XML containing one annotation for the 1st page that adds a black rectangle redaction to a redacted document:

```
<?xml version="1.0"?>
<documentAnnotations>
  <pages>
    <page id="1" pageWidth="612" pageHeight="792">
      <annotation drawType="rectangle_filled_redact"
        x_percent="0.043" y_percent="0.028"
        height_percent="0.093" width_percent="0.128" />
    </page>
  </pages>
</documentAnnotations>
```

```

    </page>
  </pages>
</documentAnnotations>

```

Image Binary Data

Some annotations add images to a document. The content of such images is provided in an XML as well. To support this, a `<documentAnnotations>` element can have a child element `<stampImages>` that can include an arbitrary number of `<stampImage>` elements, each having two required elements:

Name	Type	Description
<code>imageStampId</code>	string	ID of an image that will be used by annotations to reference to it.
<code>base64format</code>	string	Base64-encoded binary image data; this attribute uses the same format as HTML inline image does.

Example

```

<documentAnnotations>
  <pages>
    <page id="1" pageWidth="612" pageHeight="792">
      <annotation drawType="imagestamp"
        imageStampId="sample-stamp-id-1"
        x_percent="0.043" y_percent="0.028"
        height_percent="0.093" width_percent="0.128" />
    </page>
  </pages>
  <stampImages>
    <stampImage imageStampId="sample-stamp-id-1"
      base64format="data: image/png;base64,iVBORw0K...kJggg==" />
  </stampImages>
</documentAnnotations>

```

Custom Type Definitions

Color - in all attributes below "color" type means the attribute's value can either be a "transparent" keyword, meaning no color at all should be applied for that piece of a redaction, or an integer value that represents a 24-bit RGB value of a desired color. This type does not support HTML hex color codes, such as "#F0F8FF" or HTML color names such as "AliceBlue".

Draw Type Descriptions

For text-based annotations, such as a highlights or strikethroughs, the annotation will only display in the search results if the text underneath the annotation matches the search terms.

arrow

Draws a [line](#) with a triangle-shaped head. Same set of attributes, the size of an arrowhead is hard-coded and it also has the same color as the line itself. An arrowhead is drawn at the END side of the line (see [dragDirection](#) property of the line drawType).

circle_filled

Draws an ellipse that fits into a bounding rectangle. Uses the same attributes as [rectangle_filled](#).

circle_trans

Draws an ellipse that fits into a bounding rectangle. Uses the same attributes as [rectangle_filled](#).

eSignDate

Legacy synonym for [text_redact](#) annotation. > **_NOTE:** Coordinate calculations may be slightly different from [text_redact](#) annotation.

eSignEmail

Legacy synonym for [text_redact](#) annotation. > **_NOTE:** Coordinate calculations may be slightly different from [text_redact](#) annotation.

eSignESignId

Legacy synonym for [text_redact](#) annotation. > **_NOTE:** Coordinate calculations may be slightly different from [text_redact](#) annotation.

eSignInitials

Legacy synonym for [text_redact](#) annotation. > **_NOTE:** Coordinate calculations may be slightly different from [text_redact](#) annotation.

eSignName

Legacy synonym for [text_redact](#) annotation. > **_NOTE:** Coordinate calculations may be slightly different from [text_redact](#) annotation.

eSignSignature

Draws a text entry that is composed of two text strings and a date string. Allows adding current date to the document. Its content is positioned at the bottom of the bounding rectangle. Note that `eSignWidth` and `eSignHeight` attributes override `width_percent` and `height_percent` attributes of the boundary rectangle.

Annotation text is built from following parts, in that order:

1. `eSignIdLabel` attribute value;
2. `uuid` attribute value;
3. `eSignDate` attribute value. If this attribute is not specified, current date is used.

Text attributes are controlled by an esign-specific set of attributes (see below), unlike other text annotations that are controlled by HTML-like markup.

Supported attributes:

Name	Type	Required / Default	Description
<code>eSignWidth</code>	int	280	Overrides the boundary rectangle width.
<code>eSignHeight</code>	int	10	Overrides the boundary rectangle height.

Name	Type	Required / Default	Description
eSignIdLabel	string	eSign ID:	Signature label (first part of the signature annotation text).
eSignUuid	string		Signature uuid (second part of the signature annotation text).
eSignDate	string		Signature date (third part of the signature annotation text).
eSignFontFace	string	Tahoma	Font name to use; same limitations apply as when using a font name for a text_redact .
eSignFontSize	int	8	Font size to use.

See also: [Common Attributes](#)

eSignText

Legacy synonym for [text_redact](#) annotation. > **_NOTE:** coordinate calculations may be slightly different from [text_redact](#) annotation._

eSignTitle

Legacy synonym for [text_redact](#) annotation. > **_NOTE:** coordinate calculations may be slightly different from [text_redact](#) annotation._

freehand

Draws an arbitrary SVG path using a limited subset of commands - only absolute versions of [L](#), [C](#) and [M](#) SVG path commands are supported at the moment. The drawn path is scaled to fit a boundary rectangle using the parameters listed below, while keeping the aspect ratio.

The path data itself is expected to be provided in a CDATA child element of the freehand `<annotation>` element. The path data is expected to be a set of commands, each followed by a comma-separated list of coordinate arguments; see the following example:

```
<annotation drawType="freehand" align="left" lineWidth="4"
  pathWidth="381.6459330143541" pathHeight="178.622009569378"
  x_percent="0.057416267942583726" y_percent="0.10598811077279977"
  height_percent="0.22553284036537627" width_percent="0.6236044657097289">

  <![CDATA[M0,50.75L0.08,50.51C0.16,50.26,0.32,49.77,0.97,48.31]]>

</annotation>
```

Supported attributes:

Name	Type	Required / Default	Description
pathWidth	float	Yes	Absolute width of the provided path. This is expected to correlate with the coordinates provided within the path data. This parameter is used to define the scaling factor when scaling the path to fit into the boundary rectangle.
pathHeight	float	Yes	Absolute height of the provided path. This is expected to correlate with the coordinates provided within the path data. This parameter is used to define the

Name	Type	Required / Default	Description
			scaling factor when scaling the path to fit into the boundary rectangle.
<code>lineWidth</code>	int	1	Width of the path line, in pixels
<code>lineColor</code>	color	black	Color of the path line
<code>align</code>	string	No	Path line horizontal alignment in the annotation boundary rectangle. Can have one of the following self-explanatory values: <ul style="list-style-type: none"> • right • left Missing the parameter or setting it to any other value will result to default behavior (center aligned path line).

See also: [Common Attributes](#)

highlightText

This is an utility annotation that effectively allows to draw several rectangles using the same color, border and opacity settings as a [rectangle_filled](#), while having a set of rectangles defined in `<line-rectangles>` child elements exactly as it is done in [text_hyperlink_annotation](#).

imagestamp

A legacy synonym of an [imagestamp_redact](#).

imagestamp_redact

Draws an image over a given area. The image is scaled to fit its content into a boundary rectangle, keeping aspect ratio.

Supported attributes:

Name	Type	Required / Default	Description
<code>imageStampId</code>	string	Yes	id of an image to use (see Image Binary Data above).

See also: [Common Attributes](#)

line

Draws a line from one corner of a bounding rectangle to another. A line has a *drag direction* (see the attributes list); the following rules apply to a drag direction:

- if it starts with "t", then the line is directed "to the top", meaning it goes from (Y + height) to (Y) in terms of vertical direction; otherwise it goes "to the bottom";
- if it ends with "r" then the line goes "to the right", meaning it goes from (X) to (X + width); otherwise, it goes "to the left".

So, for instance, `dragDirection` "tl" means that a line will be drawn **from** a bottom-right **to** the top-left corner of a bounding rectangle.

Supported attributes:

Name	Type	Required / Default	Description
<code>lineWidth</code>	int	1	width of the line, in pixels
<code>lineColor</code>	color	black	color of the line
<code>dragDirection</code>	string	No	Defines the direction of the line. Defines the placement of the arrowhead when used for an arrow .

See also: [Common Attributes](#)

polyline

Draws a polyline. Annotation node of this type should contain coordinates of polyline points in its inner text, formatted as shown in example below:

```
<annotation drawType="polyline">
  <![CDATA[[{"x":166.4,"y":95.1},{ "x":262.07,"y":169.8},
{"x":184.4,"y":199.1}]]]>
</annotation>
```

NOTES:

- *Polyline coordinates are specified in points*
- *Boundary rectangle attributes are ignored for this annotation.*

Supported attributes:

Name	Type	Required / Default	Description
<code>lineColor</code>	color	black	Color of the line.
<code>lineWidth</code>	int	1	Width of the line, in pixels.

See also: [Common Attributes](#)

rectangle_filled

Draws a rectangle, either filled or transparent.

Supported attributes:

Name	Type	Required / Default	Description			
<code>lineColor</code>	color	0	Rectangle border color.			
<code>fillColor</code>	color	0	Rectangle fill color.			
<code>lineWidth</code>	integer	0	Width of border in pixels. <code>alpha</code>	integer	255	Alpha component that is added to fill and line colors. The default value is 255 for 100%

Name	Type	Required / Default	Description
			opacity.
opacity	integer	255	Synonym of alpha attribute name.

See also: [Common Attributes](#)

rectangle_filled_redact

Redacts a part of the document.

NOTE: This is the only annotation that not only adds new content to a page but **updates and removes some of the page's existing content**, by doing what is called a "deep redaction".

When a `drawType="rectangle_filled_redact"` is applied to a page, the following happens:

1. All text characters that are fully or partially covered by the area of the redaction get removed from the resulting document;
2. Images that have some parts of them covered under a redaction area get their content filled with black (this is always black, no matter the `fillColor` attribute) in the intersection area, effectively destroying any security-sensitive image data under the covered area;

NOTE: this feature is currently not supported for 1-bit JPEG 2000 images;

3. A rectangle of the given color gets drawn above the full extend of the redaction area; black by default.
4. Optionally, a text entry can be drawn over the redaction rectangle (usually explains the reason for the content redaction).

Supported attributes:

Name	Type	Required / Default	Description
lineWidth	int	1	A width of a bounding rectangle border.
lineColor	color	black	Border color of a drawn rectangle.
fillColor	color	black	Fill color of a drawn rectangle.
meta	string	No	A text to draw over a rectangle.
fontColor	color	white	A color of a meta text font; if no "meta" is set, this parameter does not affect anything.

See also: [Common Attributes](#)

rectangle_trans

Same behavior as `rectangle_filled`.

rectangle_trans_redact

This is effectively a `rectangle_filled` with different default values: default alpha is set to 25% opaque and default fill color is set to yellow (`#FFFF00`).

signature_path

Same behavior as [freehand](#).

signature_text

Same behavior as [text_redact](#), but without text wrapping, and with text centering vertically.

stamp

Adds rounded rectangle with centered text in it. Text is drawn exactly as for [text_redact](#) annotation with same attributes set, with the only exception that "stampSize" markup attribute is used for the font size instead of "size" attribute.

Supported markup attributes:

Name	Type	Required / Default	Description
stampSize	20	number	Font size in pixels.

See also: [text_redact](#)

See also: [Common Attributes](#)

stamp_redact

A legacy synonym for a [stamp](#).

strikethrough

This draw type applies a strikethrough annotation to the text that is specified inside *selectedText* property. Annotation that uses strikethrough property should contain `<line-rectangles>` and `<rectangles>` subelements, which define position and length of the strikethrough line.

```
<documentAnnotations>
  <pages>
    <page id="1" pageWidth="612" pageHeight="792">
      <annotation nodeId="B4C6BF9B-C220-44B9-577C-8F4ADB7EED17"
        lineColor="0" interactionMode="0" stampSize="122"
        opacity="255" x="163.01" y="167.48"
        height="18.479999999999999" width="38.94"
        drawType="strikethrough" startIndex="70"
        selectedText="True" textLength="4"
        highlightGroupID="13_1488462726580_8836" isHighlight="y"
        lineWidth="2" dragDirection="br" meta=""
        label="" saveDate="Thurs Mar 2 2017"
        saveTime="13:52:12 GMT+0000" uuid="" formUser=""
        created="Thurs Mar 2 13:52:6 GMT+0000 2017" modified="Thurs Mar 2
13:52:6 GMT+0000 2017">
        <line-rectangles>
          <rectangle x="163.01" y="167.48" height="18.479999999999999"
width="38.94"/>
        </line-rectangles>
        <![CDATA[]]>
      </annotation>
    </page>
    <page id="2" pageWidth="612" pageHeight="792"/>
  </pages>
</highlights />
```

```
</documentAnnotations>
```

Supported attributes:

Name	Type	Required / Default	Description
<code>lineWidth</code>	integer	2	Strikethrough line thickness in pixels.

text

Adds text with border. This acts exactly as applying both `rectangle_trans` and `text_redact` annotations with the attributes set that includes the attributes of both abovementioned draw types.

text_hyperlink_annotation

Adds hyperlink to the document.

Supported attributes:

Name	Type	Required / Default	Description
<code>href</code>	yes	string	HTTP URL.
<code>lineColor</code>	0	color	Base color for hyperlink highlighting. If it is set to 0 or transparent, then default value will be used, which is blue color.
<code>fillColor</code>	0	color	Reserved, should be set to 0.

See also: [Common Attributes](#)

Annotation node of this type should contain `<line-rectangles>` and `<rectangles>` subelements, which define hyperlink area. The area, containing of several rectangles, would be highlighted and clickable. Each rectangle is defined by its top left corner coordinates, width and height as shown in this example:

```
<annotation ... >
  ...
  <line-rectangles>
    <rectangle x="226.8" y="225.7" height="11.3" width="315.5"/>
    <rectangle x="72" y="238.4" height="11.3" width="183.4"/>
  </line-rectangles>
</annotation>
```

text_input_signature (reserved for internal use)

Reserved for internal use and not currently supported by the Redaction API.

text_redact

This draw type adds a text entry to a document.

There are no specific XML attributes (besides the usual bounding rectangle) for this draw type. For legacy reasons, parametrization of the text attributes (font, color, positioning and wrapping) **are set in an html-like piece of markup** in a `CDATA` child element of the `<annotation>`, like this:

```
<annotation x_percent="0.044" y_percent="0.052" height_percent="0.072"
width_percent="0.21" drawType="text_redact">
  <![CDATA[<TEXTFORMAT><P ALIGN="left" ><FONT FACE="Arial" SIZE= "12"
COLOR="#000000">Sample Text Redaction
With two lines</FONT></P></TEXTFORMAT>]]>
</annotation>
```

Supported HTML-like font markup attributes:

Name	Type	Required / Default	Description
<code>ALIGN</code>	string	No	Controls the horizontal alignment of a text entry within the bounding rectangle area. Can have one of the following self-explanatory values: <ul style="list-style-type: none"> • RIGHT • MIDDLE or CENTER (synonyms) Setting <code>ALIGN</code> to any other value will result to default behavior (left aligned text).
<code>FACE</code>	string	Arial	Name of the font to use. For purposes of this annotation, the following font names can be used: <ul style="list-style-type: none"> • a font that is available to the OS; • a font that resides in a OS-specific fonts folder; • a font that resides in a specified fonts folder of a product (usually <code>./modules/fonts</code>).
<code>SIZE</code>	float	20	Font size to use.
<code>COLOR</code>	string	#000000	A usual HTML hex representation of an RGB color of a font to use: <ul style="list-style-type: none"> • This attribute uses a HTML-like hex color notation unlike the <code>-color</code> attributes of other annotations; • HTML color names are NOT supported; • You can NOT control the opacity of a text entry, it is always 100% opaque and will remain as such even when used as part of a semi-opaque complex annotation like <code>text</code>.

`text_selection_redaction` (reserved for internal use)

Reserved for internal use and not currently supported by Redaction API.

Markup JSON Specification

Introduction

The following specification describes the PrizmDoc Markup JSON format. A formal JSON schema is also available for automated validation and can be found within the [Use the Markup JSON Schema](#) topic.

Specification

The Markup JSON format consists of a top-level object with a `marks` property that is an array of all marks related to a single document.

Example

The following Markup JSON example creates a rectangle annotation with red background and black border near the top-left corner of the first page in the document:

```
{
  "marks": [{
    "uid": "bmd30F8yMDE5LTaxLTE4VDEyOjE5OjU2LjQ0M1pfNGY3MTFh",
    "interactionMode": "Full",
    "pageNumber": 1,
    "type": "RectangleAnnotation",
    "creationDateTime": "2019-01-18T12:19:56.442Z",
    "modificationDateTime": "2019-01-18T12:19:58.089Z",
    "data": {},
    "rectangle": {
      "x": 15.622641509433958,
      "y": 15.622641509433958,
      "width": 143.99999999999997,
      "height": 103.2452830188679
    },
    "pageData": {
      "width": 612,
      "height": 792
    },
    "borderColor": "#000000",
    "borderThickness": 4,
    "fillColor": "#ff0000",
    "opacity": 255
  }
]
```

Properties

For text-based annotations, such as a highlights or strikethroughs, the annotation will only display in the search results if the text underneath the annotation matches the search terms.

- `marks` (Array of Objects) **Required.** Array of marks for a document. Each item must be an object that conforms to one of the supported marks with the following common properties:
 - `type` (String) **Required.** Type of mark. The following values are allowed:
 - "EllipseAnnotation"
 - "FreehandAnnotation"
 - "FreehandSignature"
 - "HighlightAnnotation"
 - "ImageStampAnnotation"
 - "ImageStampRedaction"
 - "LineAnnotation"
 - "PolylineAnnotation"

- "RectangleAnnotation"
- "RectangleRedaction"
- "StampAnnotation"
- "StampRedaction"
- "StrikethroughAnnotation"
- "TextAnnotation"
- "TextAreaSignature"
- "TextHyperlinkAnnotation"
- "TextInputSignature"
- "TextRedaction"
- "TextSelectionRedaction"
- "TextSignature"
- "TransparentRectangleRedaction"
- uid (String) **Required.** Unique id for this mark. This can be any globally-unique string, like a GUID.
- interactionMode (String) **Required.** Interaction mode for this mark. The following values are allowed:
 - "Full" - Indicates that the mark is fully interactive using the mouse, touch-input, and API.
 - "SelectionDisabled" - Indicates that the mark cannot be selected.
- pageNumber (Integer) **Required.** One-indexed page where this mark is located. For example, the first page of a document will be 1.
- data (Object) A property bag of user-defined values. Property values are only allowed to be strings.
- creationDateTime (String) **Required.** Date and time when this mark was created. Format is [RFC 3339 Internet Date/Time profile of ISO 8601](#), e.g. "2016-11-05T08:15:30.494Z".
- modificationDateTime (String) **Required.** Date and time when this mark was last modified. Format is [RFC 3339 Internet Date/Time profile of ISO 8601](#), e.g. "2016-11-05T08:15:30.494Z".
- conversation (Object) Comments and associated metadata for this mark.
 - data (Object) **Required.** A property bag of user-defined values. Property values are only allowed to be strings.
 - comments (Array of Objects) Array of comment objects for this mark. Items must contain:
 - data (Object) **Required.** A property bag of user-defined values. Property values are only allowed to be strings.
 - creationDateTime (String) **Required.** Date and time when this comment was created. Format is [RFC 3339 Internet Date/Time profile of ISO 8601](#), e.g. "2016-11-05T08:15:30.494Z".
 - text (String) **Required.** Text of the comment.
- *Additional properties specific to each mark type are defined in the sections below.*

Mark Specific Properties

For text-based annotations, such as a highlights or strikethroughs, the annotation will only display in the search results if the text underneath the annotation matches the search terms.

EllipseAnnotation

- rectangle (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - x (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.
 - y (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
 - width (Number) **Required.** Width of the mark bounding rectangle.
 - height (Number) **Required.** Height of the mark bounding rectangle.
- pageData (Object) **Required.** Metadata of the page where this mark is located.
 - width (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.

- `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `borderColor` (String) **Required.** Color of the border specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red border. The special value "transparent" is also allowed which will create an invisible border.
- `borderThickness` (Integer) **Required.** Thickness of the border in pixels.
- `fillColor` (String) **Required.** Color of the background specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red background. The special value "transparent" is also allowed which will create an invisible background.
- `opacity` (Integer) **Required.** Opacity of this mark. Value must be between 0 and 255. A value of 0 will create an invisible mark. A value of 255 will create a fully opaque mark.

FreehandAnnotation

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
 - `width` (Number) **Required.** Width of the mark bounding rectangle.
 - `height` (Number) **Required.** Height of the mark bounding rectangle.
- `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `color` (String) **Required.** Color of the line specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red line.
- `opacity` (Integer) **Required.** Opacity of this mark. Value must be between 0 and 255. A value of 0 will create an invisible mark. A value of 255 will create a fully opaque mark.
- `path` (String) **Required.** An SVG-style path using M, L, and C commands used to draw a line.
- `thickness` (Number) **Required.** Thickness of the line in pixels. Value must be between 1 and 50.

FreehandSignature

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
 - `width` (Number) **Required.** Width of the mark bounding rectangle.
 - `height` (Number) **Required.** Height of the mark bounding rectangle.
- `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `color` (String) **Required.** Color of the line specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red line.
- `horizontalAlignment` (String) **Required.** Horizontal alignment applied to the text of this mark. The following values are allowed:
 - "Left" - The left edge of the mark will be horizontally anchored to the left of the mark bounding rectangle.
 - "Center" - The center of the mark will be horizontally anchored to the center of the mark bounding rectangle.
 - "Right" - The right edge of the mark will be horizontally anchored to the right of the mark

bounding rectangle.

- `path` (String) **Required.** An SVG-style path using M, L, and C commands used to draw a line.
- `thickness` (Number) **Required.** Thickness of the line in pixels. Value must be between 1 and 50.

HighlightAnnotation

- `fillColor` (String) **Required.** Color of the highlight specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red highlight.
- `startIndex` (Integer) **Required.** First zero-indexed character in the first line of text selected for this mark.
- `selectedText` (String) **Required.** Text selected across all lines for this mark.
- `textLength` (Integer) **Required.** Number of characters across all lines of text selected for this mark.
- `lineGroups` (Array of Objects) **Required.** Array of objects describing the selected text on each page where this mark will be applied. Items must contain:
 - `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `pageNumber` (Integer) **Required.** One-indexed page where the text selected by this line group is located. For example, the first page of a document will be 1.
 - `startIndex` (Integer) **Required.** First zero-indexed character of the text selected on this page.
 - `textLength` (Integer) **Required.** Number of characters selected across all lines on this page.
 - `lines` (Array of Objects) **Required.** Array of objects describing each line of the selected text on this page. Items must contain:
 - `rectangle` (Object) **Required.** Bounding rectangle dimensions of the selected text on a single line.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of the selected text.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of the selected text.
 - `width` (Number) **Required.** Width of the selected text bounding rectangle.
 - `height` (Number) **Required.** Height of the selected text bounding rectangle.

ImageStampAnnotation

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
 - `width` (Number) **Required.** Width of the mark bounding rectangle.
 - `height` (Number) **Required.** Height of the mark bounding rectangle.
- `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `imageUrl` (String) **Required.** Image data for this mark. Format is [RFC 2397 data URL scheme](#), e.g. "data:image/png;base64,R0lGOD...". Only `image/png` and `image/gif` media types are supported for burning.
- `imageId` (String) **Required.** Unique id for this image. This can be any globally-unique string, like a GUID.

ImageStampRedaction

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.

- `x` (Number) **Required**. Distance from the left edge of the page to the left edge of this mark.
- `y` (Number) **Required**. Distance from the top edge of the page to the top edge of this mark.
- `width` (Number) **Required**. Width of the mark bounding rectangle.
- `height` (Number) **Required**. Height of the mark bounding rectangle.
- `pageData` (Object) **Required**. Metadata of the page where this mark is located.
 - `width` (Number) **Required**. Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required**. Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `imageUrl` (String) **Required**. Image data for this mark. Format is [RFC 2397 data URL scheme](#), e.g. `"data:image/png;base64,R0lGOD..."`. Only `image/png` and `image/gif` media types are supported for burning.
- `imageId` (String) **Required**. Unique id for this image. This can be any globally-unique string, like a GUID.

LineAnnotation

- `pageData` (Object) **Required**. Metadata of the page where this mark is located.
 - `width` (Number) **Required**. Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required**. Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `endPoint` (Object) **Required**. Ending point of this mark on the page.
 - `x` (Number) **Required**. Distance from the left edge of the page to the end point of this mark.
 - `y` (Number) **Required**. Distance from the top edge of the page to the end point of this mark.
- `startPoint` (Object) **Required**. Starting point of this mark on the page.
 - `x` (Number) **Required**. Distance from the left edge of the page to the start point of this mark.
 - `y` (Number) **Required**. Distance from the top edge of the page to the start point of this mark.
- `color` (String) **Required**. Color of the line specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of `"#FF0000"` will create a red line.
- `endHeadType` (String) **Required**. Shape drawn at the end point of the line. Allowed values are:
 - `"None"` - Indicates no shape will be drawn at the end point of the line.
 - `"FilledTriangle"` - Indicates a filled triangle will be drawn at the end point of the line which makes the line look like an arrow.
- `opacity` (Integer) **Required**. Opacity of this mark. Value must be between 0 and 255. A value of 0 will create an invisible mark. A value of 255 will create a fully opaque mark.
- `thickness` (Number) **Required**. Thickness of the line in pixels. Value must be between 1 and 50.

PolylineAnnotation

- `pageData` (Object) **Required**. Metadata of the page where this mark is located.
 - `width` (Number) **Required**. Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required**. Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `points` (Array of Objects) **Required**. Array of objects describing each point on the line of this mark. Items must contain:
 - `startPoint` (Object) **Required**. Starting point of this mark on the page.
 - `x` (Number) **Required**. Distance from the left edge of the page to the start point of this mark.
 - `y` (Number) **Required**. Distance from the top edge of the page to the start point of this mark.
- `color` (String) **Required**. Color of the line specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of `"#FF0000"` will create a red line.
- `opacity` (Integer) **Required**. Opacity of this mark. Value must be between 0 and 255. A value of 0 will

create an invisible mark. A value of 255 will create a fully opaque mark.

- `thickness` (Number) **Required.** Thickness of the line in pixels. Value must be between 1 and 50.

RectangleAnnotation

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
 - `width` (Number) **Required.** Width of the mark bounding rectangle.
 - `height` (Number) **Required.** Height of the mark bounding rectangle.
- `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `borderColor` (String) **Required.** Color of the border specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red border. The special value "transparent" is also allowed which will create an invisible border.
- `borderThickness` (Integer) **Required.** Thickness of the border in pixels.
- `fillColor` (String) **Required.** Color of the background specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red background. The special value "transparent" is also allowed which will create an invisible background.
- `opacity` (Integer) **Required.** Opacity of this mark. Value must be between 0 and 255. A value of 0 will create an invisible mark. A value of 255 will create a fully opaque mark.

RectangleRedaction

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
 - `width` (Number) **Required.** Width of the mark bounding rectangle.
 - `height` (Number) **Required.** Height of the mark bounding rectangle.
- `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `borderColor` (String) **Required.** Color of the border specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red border.
- `borderThickness` (Integer) **Required.** Thickness of the border in pixels.
- `fillColor` (String) **Required.** Color of the background specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red background.
- `fontColor` (String) **Required.** Color of the text specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create red text.
- `reasons` (String[]) or `reason` (String) **Required.** Text to display in the center of the redaction indicating why the redaction was made. You can provide either a plural `reasons` property with an array of strings or a singular `reason` property with a single string value. If you provide an array of `reasons`, they will be displayed together as a single string with a semicolon separating each reason.

StampAnnotation

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.

- `y` (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
- `width` (Number) **Required.** Width of the mark bounding rectangle.
- `height` (Number) **Required.** Height of the mark bounding rectangle.
- `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `color` (String) **Required.** Color of the stamp border and text specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red stamp.
- `label` (String) **Required.** Text aligned in the center of the mark bounding rectangle.

StampRedaction

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
 - `width` (Number) **Required.** Width of the mark bounding rectangle.
 - `height` (Number) **Required.** Height of the mark bounding rectangle.
- `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `label` (String) **Required.** Text aligned in the center of the mark bounding rectangle.

StrikethroughAnnotation

- `color` (String) **Required.** Color of the strikethrough line specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red strikethrough line.
- `thickness` (Number) **Required.** Thickness of the line in pixels. Value must be between 1 and 50.
- `textLength` (Integer) **Required.** Number of characters across all lines of text selected for this mark.
- `startIndex` (Integer) **Required.** First zero-indexed character in the first line of text selected for this mark.
- `selectedText` (String) **Required.** Text selected across all lines for this mark.
- `lineGroups` (Array of Objects) **Required.** Array of objects describing the selected text on each page where this mark will be applied. Items must contain:
 - `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `pageNumber` (Integer) **Required.** One-indexed page where the text selected by this line group is located. For example, the first page of a document will be 1.
 - `startIndex` (Integer) **Required.** First zero-indexed character of the text selected on this page.
 - `textLength` (Integer) **Required.** Number of characters selected across all lines on this page.
 - `lines` (Array of Objects) **Required.** Array of objects describing each line of the selected text on this page. Items must contain:
 - `rectangle` (Object) **Required.** Bounding rectangle dimensions of the selected text on a single line.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of the selected text.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of the

- selected text.
- `width` (Number) **Required.** Width of the selected text bounding rectangle.
- `height` (Number) **Required.** Height of the selected text bounding rectangle.

TextAnnotation

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
 - `width` (Number) **Required.** Width of the mark bounding rectangle.
 - `height` (Number) **Required.** Height of the mark bounding rectangle.
- `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `borderColor` (String) **Required.** Color of the border specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red border. The special value "transparent" is also allowed which will create an invisible border.
- `borderThickness` (Integer) **Required.** Thickness of the border in pixels.
- `fillColor` (String) **Required.** Color of the background specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red background. The special value "transparent" is also allowed which will create an invisible background.
- `fontColor` (String) **Required.** Color of the text specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create red text.
- `fontName` (String) **Required.** Font name for the text.
- `fontSize` (Number) **Required.** Font size for the text specified in pixels. Value must be between 0.1 and 288.
- `fontStyle` (String[]) **Required.** Font styling applied to the text. An array of String values. The following values are allowed:
 - "Bold"
 - "Italic"
 - "Strikeout"
 - "Underline"
- `maxLength` (Integer) **Required.** Maximum number of characters allowed. Value must be greater than or equal to 0. A value of 0 indicates a maximum length will not be enforced.
- `horizontalAlignment` (String) **Required.** Horizontal alignment applied to the text of this mark. The following values are allowed:
 - "Left" - The left edge of the mark will be horizontally anchored to the left of the mark bounding rectangle.
 - "Center" - The center of the mark will be horizontally anchored to the center of the mark bounding rectangle.
 - "Right" - The right edge of the mark will be horizontally anchored to the right of the mark bounding rectangle.
- `text` (String) **Required.** Text of this mark.
- `opacity` (Integer) **Required.** Opacity of this mark. Value must be between 0 and 255. A value of 0 will create an invisible mark. A value of 255 will create a fully opaque mark.

TextAreaSignature

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.

- `y` (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
- `width` (Number) **Required.** Width of the mark bounding rectangle.
- `height` (Number) **Required.** Height of the mark bounding rectangle.
- `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `fontColor` (String) **Required.** Color of the text specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create red text.
- `fontName` (String) **Required.** Font name for the text.
- `fontStyle` (String[]) **Required.** Font styling applied to the text. An array of String values. The following values are allowed:
 - "Bold"
 - "Italic"
 - "Strikeout"
 - "Underline"
- `horizontalAlignment` (String) **Required.** Horizontal alignment applied to the text of this mark. The following values are allowed:
 - "Left" - The left edge of the mark will be horizontally anchored to the left of the mark bounding rectangle.
 - "Center" - The center of the mark will be horizontally anchored to the center of the mark bounding rectangle.
 - "Right" - The right edge of the mark will be horizontally anchored to the right of the mark bounding rectangle.
- `maxFontSize` (Integer) **Required.** Maximum size of the font specified in pixels. Value must be greater than or equal to 0. A value of 0 indicates a maximum font size will not be enforced.
- `maxLength` (Integer) **Required.** Maximum number of characters allowed. Value must be greater than or equal to 0. A value of 0 indicates a maximum length will not be enforced.
- `text` (String) **Required.** Text of this mark.

TextHyperlinkAnnotation

- `fillColor` (String) **Required.** Color of the background specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create a red background.
- `textLength` (Integer) **Required.** Number of characters across all lines of text selected for this mark.
- `startIndex` (Integer) **Required.** First zero-indexed character in the first line of text selected for this mark.
- `selectedText` (String) **Required.** Text selected across all lines for this mark.
- `lineGroups` (Array of Objects) **Required.** Array of objects describing the selected text on each page where this mark will be applied. Items must contain:
 - `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `pageNumber` (Integer) **Required.** One-indexed page where the text selected by this line group is located. For example, the first page of a document will be 1.
 - `startIndex` (Integer) **Required.** First zero-indexed character of the text selected on this page.
 - `textLength` (Integer) **Required.** Number of characters selected across all lines on this page.
 - `lines` (Array of Objects) **Required.** Array of objects describing each line of the selected text on this page. Items must contain:
 - `rectangle` (Object) **Required.** Bounding rectangle dimensions of the selected text on a

single line.

- `x` (Number) **Required.** Distance from the left edge of the page to the left edge of the selected text.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of the selected text.
 - `width` (Number) **Required.** Width of the selected text bounding rectangle.
 - `height` (Number) **Required.** Height of the selected text bounding rectangle.
- `href` (String) **Required.** URL of the hyperlink.

TextInputSignature

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
 - `width` (Number) **Required.** Width of the mark bounding rectangle.
 - `height` (Number) **Required.** Height of the mark bounding rectangle.
- `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `mask` (Object) **Required.** Object defining the input mask for this mark. An input mask defines the format of the requested data and automatically rejects invalid input from the user. For example, the following input mask for a US telephone number allows only parentheses, dashes and numbers: "(###) ###-####". A value of `null` indicates an input mask should not be applied.
 - `value` (String) **Required.** String representation of the mask. The user input will *look* like this string once they have finished their input. Each character in this string that does not have a translation will be represented to the user literally.
 - `translations` (Object) **Required.** Translations to use for the given mask character. The key represents a character present in the mask value, and the value is a regular expression which validates the acceptable user input for that character. For example, the translation object { "#" : "/\\d/" } specifies that the "#" character in a mask string will allow only a number to be entered.
- `fontColor` (String) **Required.** Color of the text specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create red text.
- `fontName` (String) **Required.** Font name for the text.
- `maxLength` (Integer) **Required.** Maximum number of characters allowed. Value must be greater than or equal to 0. A value of 0 indicates a maximum length will not be enforced.
- `text` (String) **Required.** Text of this mark.
- `horizontalAlignment` (String) **Required.** Horizontal alignment applied to the text of this mark. The following values are allowed:
 - "Left" - The left edge of the mark will be horizontally anchored to the left of the mark bounding rectangle.
 - "Center" - The center of the mark will be horizontally anchored to the center of the mark bounding rectangle.
 - "Right" - The right edge of the mark will be horizontally anchored to the right of the mark bounding rectangle.

TextRedaction

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
 - `width` (Number) **Required.** Width of the mark bounding rectangle.

- `height` (Number) **Required.** Height of the mark bounding rectangle.
- `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `fontColor` (String) **Required.** Color of the text specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create red text.
- `fontName` (String) **Required.** Font name for the text.
- `fontSize` (Number) **Required.** Font size for the text specified in pixels. Value must be between 0.1 and 288.
- `maxLength` (Integer) **Required.** Maximum number of characters allowed. Value must be greater than or equal to 0. A value of 0 indicates a maximum length will not be enforced.
- `horizontalAlignment` (String) **Required.** Horizontal alignment applied to the text of this mark. The following values are allowed:
 - "Left" - The left edge of the mark will be horizontally anchored to the left of the mark bounding rectangle.
 - "Center" - The center of the mark will be horizontally anchored to the center of the mark bounding rectangle.
 - "Right" - The right edge of the mark will be horizontally anchored to the right of the mark bounding rectangle.
- `text` (String) **Required.** Text of this mark.

TextSelectionRedaction

- `lineGroups` (Array of Objects) **Required.** Array of objects describing the selected text on each page where this mark will be applied. Items must contain:
 - `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `pageNumber` (Integer) **Required.** One-indexed page where the text selected by this line group is located. For example, the first page of a document will be 1.
 - `startIndex` (Integer) **Required.** First zero-indexed character of the text selected on this page.
 - `textLength` (Integer) **Required.** Number of characters selected across all lines on this page.
 - `lines` (Array of Objects) **Required.** Array of objects describing each line of the selected text on this page. Items must contain:
 - `rectangle` (Object) **Required.** Bounding rectangle dimensions of the selected text on a single line.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of the selected text.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of the selected text.
 - `width` (Number) **Required.** Width of the selected text bounding rectangle.
 - `height` (Number) **Required.** Height of the selected text bounding rectangle.
- `reasons` (String[]) or `reason` (String) **Required.** Text to display in the center of the redaction indicating why the redaction was made. You can provide either a plural `reasons` property with an array of strings or a singular `reason` property with a single string value. If you provide an array of `reasons`, they will be displayed together as a single string with a semicolon separating each reason.
- `selectedText` (String) **Required.** Text selected across all lines for this mark.
- `startIndex` (Integer) **Required.** First zero-indexed character in the first line of text selected for this mark.

- `textLength` (Integer) **Required.** Number of characters across all lines of text selected for this mark.

TextSignature

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
 - `width` (Number) **Required.** Width of the mark bounding rectangle.
 - `height` (Number) **Required.** Height of the mark bounding rectangle.
- `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
- `color` (String) **Required.** Color of the text specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of "#FF0000" will create red text.
- `fontName` (String) **Required.** Font name for the text.
- `horizontalAlignment` (String) **Required.** Horizontal alignment applied to the text of this mark. The following values are allowed:
 - "Left" - The left edge of the mark will be horizontally anchored to the left of the mark bounding rectangle.
 - "Center" - The center of the mark will be horizontally anchored to the center of the mark bounding rectangle.
 - "Right" - The right edge of the mark will be horizontally anchored to the right of the mark bounding rectangle.
- `text` (String) **Required.** Text of this mark.

TransparentRectangleRedaction

- `rectangle` (Object) **Required.** Bounding rectangle dimensions of this mark on the page.
 - `x` (Number) **Required.** Distance from the left edge of the page to the left edge of this mark.
 - `y` (Number) **Required.** Distance from the top edge of the page to the top edge of this mark.
 - `width` (Number) **Required.** Width of the mark bounding rectangle.
 - `height` (Number) **Required.** Height of the mark bounding rectangle.
- `pageData` (Object) **Required.** Metadata of the page where this mark is located.
 - `width` (Number) **Required.** Width of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.
 - `height` (Number) **Required.** Height of the page, in pixels, at the time the mark is saved. Value must be greater than or equal to 0.

How To Examples

This section contains the following "How To" information:

- [Compare Documents](#)
- [Convert Content with Content Conversion Service](#)
 - [Content Conversion Demo](#)
 - [How to Configure the Demo on Windows](#)
 - [How to Configure the Demo on Linux](#)
- [Migrate from PrizmDoc Cloud Servers to PrizmDoc Viewer Self-Hosted Servers](#)
- [Perform Auto-Redaction](#)
- [Set up a Viewing Session for a CAD Drawing which has XREF Dependencies](#)

- [Use the Markup JSON Schema](#)
- [Use a Viewing Session](#)
- [Watermark Content in a Viewing Session](#)

Compare Documents

Introduction

The Document Comparison feature compares two documents and outputs a document that indicates the differences between the two, such as formatting changes, grammatical changes, or the addition or omission of content.

How-To Overview

To compare Microsoft Word documents:

1. [Upload the original document](#) used in a comparison viewing session, as well as the [revised document](#) used in a comparison viewing session.
2. [Start the viewing session](#) to trigger the comparison process and initiate the view of the differences. If the original and revised documents have many pages and produce a large number of differences, the comparison process could take more time than the default timeout values are configured for. In order to let the long-lasting comparison process complete successfully [the DocumentAcquisitionTimeout and DocumentInteractiveTimeout timeouts](#) should be increased accordingly. The default timeout values allow you to compare documents having less than 100 pages and less than 1000 differences.
3. Once the viewing session is initiated, you can start issuing API [requests to obtain revision data](#) from a viewing session that compares documents. If the result of comparing two documents contains thousands of differences, in order to obtain all of them [the InternalOperationTimeout timeout](#) should be increased accordingly. The default timeout value allows you to obtain up to 1000 revisions per session.

PCCIS Timeouts Configuration Examples

Depending of the complexity and the size of the documents being compared, the comparison process could take more time than the default [PCCIS timeouts](#) are configured for. Please consider the following examples below when comparing such documents and adjust the PCCIS timeouts accordingly.

When comparing documents having less than 100 pages and producing less than 1000 differences:

- `<DocumentInteractiveTimeout>50000</DocumentInteractiveTimeout>`
- `<DocumentAcquisitionTimeout>45000</DocumentAcquisitionTimeout>`
- `<InternalOperationTimeout>100000</InternalOperationTimeout>`

When comparing documents having less than 300 pages and producing less than 3000 differences:

- `<DocumentInteractiveTimeout>70000</DocumentInteractiveTimeout>`
- `<DocumentAcquisitionTimeout>70000</DocumentAcquisitionTimeout>`
- `<InternalOperationTimeout>350000</InternalOperationTimeout>`

When comparing documents having less than 600 pages and producing less than 5000 differences:

- `<DocumentInteractiveTimeout>90000</DocumentInteractiveTimeout>`
- `<DocumentAcquisitionTimeout>90000</DocumentAcquisitionTimeout>`
- `<InternalOperationTimeout>500000</InternalOperationTimeout>`

When comparing documents having more than 600 pages and producing more than 5000 differences:

- `<DocumentInteractiveTimeout>100000</DocumentInteractiveTimeout>`
- `<DocumentAcquisitionTimeout>100000</DocumentAcquisitionTimeout>`
- `<InternalOperationTimeout>600000</InternalOperationTimeout>`

Licensing

The Microsoft Word document (.DOC and .DOCX) comparison feature uses the [Microsoft Office Conversion \(MSO\) add-on option](#) for PrizmDoc Server running on Windows and provides [connectivity](#) for PrizmDoc Server running on Linux. Therefore, this functionality is triggered by the license key that includes the MSO feature. The MSO feature can be purchased in addition to your standard PrizmDoc Server license. See [Feature Licensing](#) for more information.

Requirements

The Microsoft Word document comparison feature utilizes Microsoft Office rendering capabilities and therefore requires the components listed in the [Windows Requirements](#) section to be available on the system. Please follow all the required [Windows Installation](#) steps to let the PrizmDoc Server installer successfully pre-configure the system. When running on a non-pre-configured system, PrizmDoc Server installer for Windows will determine the required registry configuration settings specific to your server characteristics, make those changes in the registry, and require a system reboot.

Convert Content with Content Conversion Service

This section describes how to use the PrizmDoc Server *content converters* REST API and provides examples of the kinds of operations you can perform with it.

For application development in .NET, we recommend using the [PrizmDoc Server .NET SDK](#) instead of using the PrizmDoc Server REST API directly. See the [.NET SDK How to Guides](#) for examples of how to perform file conversion, OCR, document merging, and more with the .NET SDK.

The following steps walk you through using the PrizmDoc Server *content converters* REST API:

- [Step 1: Upload Your Source Document](#)
- [Step 2: Start the Content Conversion Process](#)
- [Step 3: Check Status of the ContentConverter Resource](#)
- [Step 4: Download the Converted Document\(s\)](#)
- [Conversion Input Examples](#)
 - [Multipage Word Document to Multipage PDF](#)
 - [Multipage Password Protected Word Document to Multipage PDF](#)
 - [Single-page Word Document to Scaled PNG](#)
 - [Multipage Word Document to Multiple PNG Images](#)
 - [JPEG to PNG](#)
 - [JPEG to Indexed TIFF](#)
 - [PDF to Bitonal TIFF](#)
 - [Single-page document to TIFF with specific resolution](#)
 - [Specific Pages of Two Multipage Documents to Multipage TIFF](#)
 - [All Pages of Three Multipage Documents Including Password Protected Document to Multipage PDF](#)
 - [Positioning and Text Justification within Header and Footer](#)
 - [Dynamic Page Numbering and Page Count with Optional Zero Padding within Header and Footer](#)
 - [Bates Numbering Across Multiple Output Documents](#)

- [Raster Document to a Searchable PDF](#)

Step 1: Upload Your Source Document

- Upload the source document that you want to convert.
- This can be a document of any format supported by the PrizmDoc RESTful Web Services.
- In response to this request you will receive a file ID that is used to reference the source document in later requests.

Example

```
POST http://192.168.0.1:18681/PCCIS/V1/WorkFile?FileExtension=doc
Content-Type: application/octet-stream
[binary data]

200 OK
Content-Type: application/json
{
  "fileId": "5qTYa3gzN9gYUb5SzqUhqg",
}
```

Step 2: Start the Content Conversion Process

- Using the file ID you obtained for the source document in [Step 1](#), you can now start the process to convert the document. This is accomplished by sending a POST request which will start a process that runs asynchronously on the PrizmDoc Server to produce the converted document(s).
- Specify in the POST request the output format you wish to convert the source document to. This format may be SVG, JPEG, PNG, TIFF, or PDF.
- For raster output formats (JPEG, TIFF, PNG), you may optionally specify a `maxWidth` and/or a `maxHeight`. If either of these attributes is present then the output document will be scaled to fit them as closely as possible while maintaining its original aspect ratio. At least one of `maxWidth` and `maxHeight` must be specified if the source document is in a vector format.
- If output format is PDF or TIFF, you may specify whether to convert each page of the source document to a separate output file or to convert all pages to a single document. This attribute is optional with the default value set to convert all pages to a single document.
- If input format is PDF, MS Word, MS Excel, MS PowerPoint or OpenDocument, you may optionally specify a password.

Example

```
POST http://192.168.0.1:18681/v2/contentConverters
Content-Type: application/json
{
  "input": {
    "sources": [
      {
        "fileId": "5qTYa3gzN9gYUb5SzqUhqg"
      }
    ],
    "dest": {
      "format": "pdf",
      "pdfOptions": {
```

```
        "forceOneFilePerPage": true
      }
    }
  }
}

200 OK
Content-Type: application/json
{
  "processId": "bQpcuixhvGmNqn5ElskO6Q",
  "expirationDateTime": "2014-12-03T18:30:49.460Z",
  "input": {
    "sources": [
      {
        "fileId": "5qTYa3gzN9gYUb5SzqUhqg",
        "pages": ""
      }
    ],
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": true
      }
    }
  },
  "state": "processing",
  "percentComplete": 0
}
```

Step 3: Check Status of the ContentConverter Resource

- The process to generate a converted document(s) runs asynchronously on the PrizmDoc Server. The POST request you sent in [Step 2](#) will return immediately and before the output is ready. This means you will need to check the status of the process by sending a GET request to the resource you just created.
- In response to this request, JSON will be returned that includes a `state` property. When this property is "complete", the JSON response will also include an `output` property which means you can proceed to the next step.
- See the [Content Converter API](#) for more details of this request.

Example

```
GET http://192.168.0.1:18681/v2/contentConverters/bQpcuixhvGmNqn5ElskO6Q

200 OK
Content-Type: application/json
{
  "processId": " bQpcuixhvGmNqn5ElskO6Q ",
  "expirationDateTime": "2014-12-03T18:30:49.460Z",
  "input": {
    "sources": [
      {
        "fileId": "5qTYa3gzN9gYUb5SzqUhqg",
        "pages": ""
      }
    ],
    "dest": {
```

```
    "format": "pdf",
    "pdfOptions": {
      "forceOneFilePerPage": true
    }
  },
  "state": "complete",
  "percentComplete": 100,
  "output": {
    "results": [
      {
        "fileId": "ek5Zb123oYHSUEVx1bUrVQ",
        "sources": [
          {
            "fileId": "5qTYa3gzN9gYUb5SzqUhqg",
            "pages": "1"
          }
        ],
        "pageCount": 1
      },
      {
        "fileId": "KOrSwaqsguevJ97BdmUbXi",
        "sources": [
          {
            "fileId": "5qTYa3gzN9gYUb5SzqUhqg",
            "pages": "2"
          }
        ],
        "pageCount": 1
      },
      {
        "fileId": "o349chskqw93kwaqsgfevJ",
        "sources": [
          {
            "fileId": "5qTYa3gzN9gYUb5SzqUhqg",
            "pages": "3"
          }
        ],
        "pageCount": 1
      }
    ]
  }
}
```

Step 4: Download the Converted Document(s)

- Once the content conversion process completes successfully, the new, converted document(s) are available for download.
- A work file ID is made available for each successfully converted document in the `output` property from the JSON response retrieved in [Step 3](#).
- See the [Work File API](#) for more details about downloading work files.

Example

```
GET http://192.168.0.1:18681/PCCIS/V1/WorkFile/ek5Zb123oYHSUEVx1bUrVQ
200 OK
```

```
Content-Type: application/pdf
[binary data]
```

Conversion Input Examples

Below are example JSON strings that can be used as input in [Step 2](#) above to create various ContentConverter processes.

Multipage Word Document to Multipage PDF

This example will convert all pages of a Word document to a single PDF document:

```
"input": {
  "sources": [
    {
      "fileId": "5qTYa3gzN9gYUb5SzqUhqg"
    }
  ],
  "dest": {
    "format": "pdf",
    "pdfOptions": {
      "forceOneFilePerPage": false
    }
  }
}
```

Multipage Password Protected Word Document to Multipage PDF

This example will convert all pages of a password protected Word document to a single PDF document:

```
"input": {
  "sources": [
    {
      "fileId": "5qTYa3gzN9gYUb5SzqUhqg",
      "password": "secret"
    }
  ],
  "dest": {
    "format": "pdf",
    "pdfOptions": {
      "forceOneFilePerPage": false
    }
  }
}
```

Single-page Word Document to Scaled PNG

This will convert a single page Word Document to a PNG image, scaled to 800 pixels width. Height will adjust automatically to maintain aspect ratio:


```
"input": {
  "sources": [
    {
      "fileId": "5qTYa3gzN9gYUb5SzqUhqg"
    }
  ],
  "dest": {
    "format": "png",
    "pngOptions": {
      "maxWidth": "800px"
    }
  }
}
```

Multipage Word Document to Multiple PNG Images

This will convert a multipage Word Document to multiple, single page PNG images. As PNG is not a multipage format, each page of the Word Document will be converted to a separate PNG:

```
"input": {
  "sources": [
    {
      "fileId": "5qTYa3gzN9gYUb5SzqUhqg"
    }
  ],
  "dest": {
    "format": "png"
  }
}
```

JPEG to PNG

This example will convert a JPEG image to a PNG image, scaled to fit within 800 pixels width and 600 pixels height. The output PNG will be as large as it can be while maintaining aspect ratio and remaining within these bounds:

```
"input": {
  "sources": [
    {
      "fileId": "5qTYa3gzN9gYUb5SzqUhqg"
    }
  ],
  "dest": {
    "format": "png",
    "pngOptions": {
      "maxWidth": "800px",
      "maxHeight": "600px"
    }
  }
}
```

JPEG to Indexed TIFF

This example will convert a JPEG image to TIFF image with indexed 8-bits per pixel (256) colors. The output TIFF will have some minor quality loss and its size is usually several times smaller if it is created without the color.mode: "indexed" option:

```
"input": {
  "sources": [
    {
      "fileId": "5qTYa3gzN9gYUb5SzqUhqg"
    }
  ],
  "dest": {
    "format": "tiff",
    "tiffOptions": {
      "color": {
        "mode": "indexed"
      }
    }
  }
}
```

PDF to Bitonal TIFF

This example will convert a PDF to a bitonal (black and white, 1-bit per pixel) TIFF using Group 4 compression:

```
"input": {
  "sources": [
    {
      "fileId": "5qTYa3gzN9gYUb5SzqUhqg"
    }
  ],
  "dest": {
    "format": "tiff",
    "tiffOptions": {
      "compression": {
        "type": "group4"
      },
      "color": {
        "mode": "bitonal"
      }
    }
  }
}
```

Single-page document to TIFF with specific resolution

This example will convert a single-page document to a TIFF with default compression and color mode using specific resolution (in dots per inch):

```
"input": {
  "sources": [
```

```
    {
      "fileId": "5qTYa3gzN9gYUb5SzqUhqg"
    }
  ],
  "dest": {
    "format": "tiff",
    "tiffOptions": {
      "compression": {
        "type": "auto"
      },
      "color": {
        "mode": "auto"
      },
      "dpi": 300
    }
  }
}
```

Specific Pages of Two Multipage Documents to Multipage TIFF

This example will merge first page of a Word document with the second and third pages of a PDF document and convert them to a single TIFF document:

```
"input": {
  "sources": [
    {
      "fileId": "drxx_2sNVu9VIZTS4VH2Dg",
      "pages": "1"
    },
    {
      "fileId": "qkMQmjk6CxSzt5UEY-UdFQ",
      "pages": "2-3"
    }
  ],
  "dest": {
    "format": "tiff"
  }
}
```

All Pages of Three Multipage Documents Including Password Protected Document to Multipage PDF

This example will merge together all pages of the first PDF document with all pages of the second password protected Word document and all pages of the third TIFF document and convert them to a single PDF document.

```
"input": {
  "sources": [
    {
      "fileId": "TP4TX_SxCNF86suTfHHFSw"
    },
    {
      "fileId": "oJo8CWXAqFJ0dns8UF_AzQ",
      "password": "secret"
    }
  ]
}
```

```

    },
    {
      "fileId": "EYsfBhL0JbYgNk80sbnxEg"
    }
  ],
  "dest": {
    "format": "pdf"
  }
}

```

Positioning and Text Justification within Header and Footer

Multi-dimensional array of lines indicates positioning and text justification of a header or footer content.

To put an address in the top left of every page, you can use a header with lines like this:

```

: {
  "sources": [
    {
      "fileId": "EYsfBhL0JbYgNk80sbnxEg"
    }
  ],
  "dest": {
    "format": "pdf",
    "header": {
      "lines": [
        [ "Accusoft", "", "" ],
        [ "4001 N Riverside Dr", "", "" ],
        [ "Tampa, FL 33603", "", "" ]
      ],
      "fontFamily": "Courier",
      "fontSize": "12pt",
      "color": "#F57B20"
    }
  }
}

```

By placing the text in the center position of the inner array, it will be positioned in the center of the page. For example, to print CONFIDENTIAL centered at the bottom of every page, you can define a footer with lines like this:

```

: {
  "sources": [
    {
      "fileId": "EYsfBhL0JbYgNk80sbnxEg"
    }
  ],
  "dest": {
    "format": "pdf",
    "footer": {
      "lines": [
        [ "", "CONFIDENTIAL", "" ]
      ],
      "fontFamily": "Courier",
      "fontSize": "12pt",
      "color": "#F57B20"
    }
  }
}

```

```

    }
  }
}

```

Use the following example to apply header and footer in a single call:

```

: {
  "sources": [
    {
      "fileId": "EYsfBhL0JbYgNk80sbnxEg"
    }
  ],
  "dest": {
    "format": "pdf",
    "header": {
      "lines": [
        [ "Accusoft", "", "" ],
        [ "4001 N Riverside Dr", "", "" ],
        [ "Tampa, FL 33603", "", "" ]
      ],
      "fontFamily": "Courier",
      "fontSize": "12pt",
      "color": "#F57B20"
    },
    "footer": {
      "lines": [
        [ "", "CONFIDENTIAL", "" ]
      ],
      "fontFamily": "Courier",
      "fontSize": "12pt",
      "color": "#F57B20"
    }
  }
}

```

Dynamic Page Numbering and Page Count with Optional Zero Padding within Header and Footer

You can insert the current page number and/or total page count into header or footer text using the special syntax `{{pageNumber}}` or `{{pageCount}}`.

For example, to produce a footer showing "Page 1 of 12" for the first page of a twelve-page document, you can define a footer with lines like this:

```

: {
  "sources": [
    {
      "fileId": "EYsfBhL0JbYgNk80sbnxEg"
    }
  ],
  "dest": {
    "format": "pdf",
    "footer": {
      "lines": [

```

```

        [ "", "Page {{pageNumber}} of {{pageCount}}", "" ]
    ],
    "fontFamily": "Courier",
    "fontSize": "12pt",
    "color": "#F57B20"
}
}
}

```

You can optionally pad page number and total page count values with zeroes to guarantee that they fit a particular character width using the syntax `{{pageNumber,n}}` or `{{pageCount,n}}`, where `n` is the minimum character width. If the actual page number or page count value does not meet the minimum character width, it will be left-padded with zeroes. This can be useful for bates numbering.

For example, the following code would produce a header with "Jones000097" in the top left of page 97:

```

"input": {
  "sources": [
    {
      "fileId": "EYsfBhL0JbYgNk80sbnxEg"
    }
  ],
  "dest": {
    "format": "pdf",
    "header": {
      "lines": [
        [ "Jones{{pageNumber,6}}", "", "" ]
      ],
      "fontFamily": "Courier",
      "fontSize": "12pt",
      "color": "#F57B20"
    }
  }
}
}

```

Bates Numbering Across Multiple Output Documents

You can apply Bates numbering to multiple output documents, continuing the numbering across the documents. You can do this by calculating the count of pages in already converted documents and then passing this count as a page number offset for the next conversion. Specify the offset using the syntax `{{pageNumber+c}}` where `c` is an integer constant.

The total number of pages for a converted document can be obtained from `output.results[n].pageCount` field of the response body returned for successfully completed conversion. Here is an example response where page count of converted document is equal to 15:

```

{
  "input": {
    "dest": {
      "format": "pdf"
    },
    "sources": [
      {
        "fileId": "px4x3scw_8OqzZlM24tmnQ",
        "pages": "1-15"
      }
    ]
  }
}

```

```

    }
  ]
},
"expirationDateTime": "2017-03-24T15:22:02.532Z",
"processId": "kQVvYfCtmatmWzigemW8Xw",
"state": "complete",
"percentComplete": 100,
"output": {
  "results": [
    {
      "fileId": "ZLa9F-Jg7M5gq1Wgx82ejg",
      "sources": [
        {
          "fileId": "px4x3scw_8OqzZlM24tmnQ",
          "pages": "1-15"
        }
      ],
      "pageCount": 15
    }
  ]
}
}
}

```

See the [Content Converter API](#) for more details of this.

You can optionally pad the result with zeroes using the syntax `{{pageNumber+c,n}}`, where `n` is the minimum character width. If the actual page number value does not meet the minimum character width, it will be left-padded with zeroes.

For example, if you have already converted a document containing 15 pages, and want to continue the numbering in the next conversion, using 8-digit padding, you can define a footer with lines like this:

```

"input": {
  "sources": [
    {
      "fileId": "EYsfBhL0JbYgNk80sbnxEg"
    }
  ],
  "dest": {
    "format": "pdf",
    "footer": {
      "lines": [
        [ "", "{{pageNumber+15,8}}", "" ]
      ],
      "fontFamily": "Courier",
      "fontSize": "12pt",
      "color": "#F57B20"
    }
  }
}
}

```

Raster Document to a Searchable PDF

This example will perform optical character recognition (OCR) to convert a raster file to a searchable PDF document. The resulting PDF document will contain the original image and the recognized text in a separate invisible layer, with each text character position matching its image counterpart. This will allow you to search, select

and copy the text in the resulting PDF document. > **NOTE:** *If you are attempting to make a searchable PDF from an existing PDF document, please note that the source PDF file should be an image-only PDF. PrizmDoc will not create a searchable file from already-existing vector content.*

```
"input": {
  "sources": [
    {
      "fileId": "LtrN8HwBiQOaKXvCcn9J8Q"
    }
  ],
  "dest": {
    "format": "pdf",
    "pdfOptions": {
      "ocr": {
        "language": "english"
      }
    }
  }
}
```

Content Conversion Demo

This section contains the following information:

- [How to Configure the Demo on Windows](#)
- [How to Configure the Demo on Linux](#)

How to Configure the Demo on Windows

Demo Configuration

After the installation, test the sample application in a browser:

- The Demo can be Launched by clicking **Conversion Sample** under **Accusoft | PrizmDoc Viewer** on the **Start Menu**
- The following will route you directly to the CCS Sample test page: <http://localhost:18001>

The demo.config file can be updated to customize the application host address, application port, and services port.

Example

```
{
  "logging": {
    "consoleLogFilePath":
      "%ALLUSERSPROFILE%\Accusoft\Prizm\Logs\CssDemoService.console.log"
  },
  "httpService": {
    "port": 3000
  },
  "apiService": {
    "port": 18681,
    "host": "localhost"
  }
}
```



```

    }
}

```

Demo Directory Structure

File \ Folder	Description
app.js	This is a node-js app which hosts the express app and proxy's service requests.
proxyFilter.js	This filters requests to the conversion service and white-lists headers to and from the service.
package.json	This is the node-js package json file.
node_modules	These are the node-js npm dependencies.
lib\config.js	This object reads the configuration file used to define the application port and service host and port.
demo.config	This is a configuration file used to define the application port and service host and port.
public\app.js	This contains angular js app controller, main service and some helper directives.
public\app.css	This is the application CSS.
public\index.js	This is the entry point to the app and container for the child views.
public\uploadView\upload.module.js	This is the angular js controller for the upload functionality.
public\requestCfgView\requestCfg.html	This is the markup for the CCS request configuration view.
public\requestCfgView\requestCfg.module.js	This is the controller for the CCS request configuration view.
public\requestView\request.html	This is the markup for the request configuration.
public\requestView\request.module.js	This is the controller for the request configuration.
public\conversionView\conversion.html	This is the markup for the request button.
public\conversionView\conversion.module.js	This is the controller for the request button.
public\conversionDataView\conversionData.html	This is the markup for the conversion request\response view.
public\conversionDataView\conversionData.module.js	This is the controller for the conversion request\response view.
public\downloadView\download.html	This is the markup for the download view.
public\downloadView\download.module.js	This is the controller for the download view.

Starting & Stopping the Demo

On Windows, the PrizmDemo service should ideally be started/stopped from the Windows service management console. As part of the PrizmDoc Viewer installation, the service should be configured to start automatically. If you need to start, stop, or restart, use the following instructions:

1. Log onto the system using an account with administrator privileges.
2. Click **Start > Run**.
3. Type **services.msc**.
4. Press **Enter**.
5. Find **PrizmDemo** in the list of services, and right-click on the service to access the context menu.
6. To Start the Service: Click **Start** and wait for the service to start. The status should update to "started" (this option will only be available if the service is not running).
7. To Stop the Service: Click **Stop** in the right-click menu and wait for the service control dialog. The status will be updated to be blank (this option will only be available if the service is already started).
8. To Restart the Service: Click **Restart** and wait for the service control dialog. (This option will only be available if the service is already started.)

If access to the control panel is not available, services can also be started/stopped from the command line using the following commands:

Example

```
net start PrizmDemp
net stop PrizmDemo
```

The PrizmDemo Windows service will log certain status messages to the Windows Event Log. These messages can be helpful in diagnosing problems while starting and stopping the service. To view the Windows Event Log, use the following instructions:

1. Log onto the system using an account with administrator privileges.
2. Click **Start > Run**.
3. Type **eventvwr**.
4. Press **Enter**.
5. Expand Applications and Services Logs.
6. Click **PrizmDemo**.

How to Configure the Demo on Linux

Demo Configuration

The demo.config file can be updated to customize the application host address, application port, and services port:

Example

```
{
  "logging": {
    "consoleLogFilePath":
    "/usr/share/prizm/logs/CcsDemoService.console.log"
  },
  "httpService": {
    "port": 18001
  },
  "apiService": {
```

```

    "port": 18681,
    "host": "localhost"
  }
}

```

Demo Sample Directory Structure

File / Folder	Description
app.js	This is a node-js app which hosts the express app and proxy's service requests.
proxyFilter.js	This filters requests to the conversion service and white-lists headers to and from the service.
package.json	This is the node-js package json file.
node_modules	These are the node-js npm dependencies.
lib/config.js	This object reads the configuration file used to define the application port and service host and port.
demo.config	This is a configuration file used to define the application port and service host and port.
public/app.js	This contains angular js app controller, main service and some helper directives.
public/app.css	This is the application CSS.
public/index.js	This is the entry point to the app and container for the child views.
public/uploadView/upload.module.js	This is the angular js controller for the upload functionality.
public/requestCfgView/requestCfg.htm	This is the markup for the CCS request configuration view.
public/requestCfgView/requestCfg.module.js	This is the controller for the CCS request configuration view.
public/requestView/request.html	This is the markup for the request configuration.
public/requestView/request.module.js	This is the controller for the request configuration.
public/conversionView/conversion.html	This is the markup for the request button.
public/conversionView/conversion.module.js	This is the controller for the request button.
public/conversionDataView/conversionData.html	This is the markup for the conversion request/response view.
public/conversionDataView/conversionData.module.js	This is the controller for the conversion request/response view.
public/downloadView/download.html	This is the markup for the download view.
public/downloadView/download.module.js	This is the controller for the download view.

Starting & Stopping the Demo

To start and stop the demo, use the script found in `/usr/share/prizm/scripts/demos.sh`.

Starting the demo from the shell:

Example

```
$ /usr/share/prizm/scripts/demos.sh start
```

Stopping the demo from the shell:

Example

```
$ /usr/share/prizm/scripts/demos.sh stop
```

Using the Demo

Once the demo has started, visit <http://localhost:1800> in a browser.

Migrate from PrizmDoc Cloud Servers to PrizmDoc Viewer Self-Hosted Servers

Introduction

This topic covers the steps you need to migrate from PrizmDoc Cloud Servers to PrizmDoc Viewer Self-Hosted Servers.

Step 1: Install PrizmDoc Server, PrizmDoc Viewer and PAS

1. First, [install PrizmDoc Server](#) (if you do not have it installed already).
2. Next, [install PrizmDoc Viewer and PAS](#).

IMPORTANT: Your PrizmDoc Cloud API key is not the same thing as a PrizmDoc Viewer license. Without a license, the product will run in evaluation mode with a fixed feature set. For more information about obtaining a license, please contact info@accusoft.com.

Step 2: Configure PAS to use your new PrizmDoc Server

Now that you have PrizmDoc Server installed, you have to point your PrizmDoc Application Services (PAS) instance to your new PrizmDoc server as follows:

1. Find your PAS configuration file. If you do not know where it is located, refer to the section "Configuration File Location" in the topic [PrizmDoc Application Services \(PAS\) Configuration](#).
2. Next, you will need to modify the PrizmDoc Server connection settings within your PAS configuration file. For instructions, refer to the section, "Configuring the PrizmDoc Server Connection" in the topic [PrizmDoc Application Services \(PAS\) Configuration](#).
3. Remove/comment the unnecessary `pccServer.apiKey` setting.
4. Restart PAS (see [Starting & Stopping PrizmDoc Application Services](#)).

You are done! Your PrizmDoc Application Services are now pointing to your new instance of PrizmDoc Server.

Perform Auto-Redaction

Introduction

An auto-redaction is a multi-step process that finds matches of a given regular expression, then permanently removes the text and blacks out the displayed region for each match. The end result is a new PDF document with no traces of the text that matched the regular expression.

For application development in .NET, we recommend using the [PrizmDoc Server .NET SDK](#) instead of using the PrizmDoc Server REST API directly. See the [How to Create a Redacted PDF](#) topic in the .NET SDK documentation for an example of how to easily perform auto-redaction with the .NET SDK.

The following steps walk you through using the PrizmDoc Server REST API to perform auto-redaction.

Step 1: Upload Your Source Document

- Upload the **source document** that you want to redact.
- This can be a document of any format supported by the PrizmDoc Server RESTful API, except for DICOM and CAD documents, which are not currently supported for redaction.
- In response to this request, you will receive a file ID that is used to reference the source document in later requests.

Example

```
POST /PCCIS/V1/WorkFile?FileExtension=pdf
Content-Type: application/octet-stream
[binary data]

200 OK
Content-Type: application/json
{
  "fileId": "5qTYa3gzN9gYUb5SzqUhqg",
}
```

Step 2: Compose a Regular Expression

- Compose the **regular expression** that will match the text you want to redact in the document.
- The regular expression should adhere to the POSIX extended RE (ERE) or basic RE (BRE) syntax. (See details in this link: <http://laurikari.net/tre/documentation/regex-syntax/>) > **NOTE:** *Undocumented regex features may work, however we don't provide support for them.*
- For example, the following regular expression will redact all US Social Security Numbers in a document:

Example

```
"[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"
```

NOTE: The regular expression is sent to PrizmDoc in JSON format, so you should adjust the regular expression according to JSON syntax. Specifically, the backslash symbol should be duplicated. If you create regular expressions programmatically, using string literals, you may need to further adjust the string according to the programming language syntax.

Example

```
Regular expression (searches whole word "the", case insensitive):  
"(?i:\bthe\b)"
```

```
JSON content:  
"regex": "(?i:\\bthe\\b)";
```

```
C# code:  
string regex = "(?i:\\\\bthe\\\\b)";
```

Step 3: Create Markup JSON from the Regular Expression

Before the actual redaction process can be started, the regular expression needs to be converted to a format it can understand. PrizmDoc uses a proprietary XML syntax to define markups used for redaction, which you can generate by sending a POST request that requires two inputs:

- The **file ID** of source document you uploaded in Step 1.
- One or more **rules** to match and redact the document text:
 - Each rule includes a **regular expression** such as the one you created in Step 2, and
 - An object that describes how to redact the matching text.

Example

```
POST /v2/redactionCreators  
Content-Type: application/json  
{  
  "input": {  
    "source": {  
      "fileId": "5qTYa3gzN9gYUb5SsqUhqg"  
    },  
    "rules": [{  
      "find": {  
        "type": "regex",  
        "pattern": "[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"  
      },  
      "redactWith": {  
        "type": "RectangleRedaction",  
        "reason": "Redacted"  
      }  
    }  
  }  
}  
  
200 OK
```

```
Content-Type: application/json
{
  "processId": "Rr64ma-U_HseoPrs6y0iiw",
  "expirationDateTime": "2014-12-03T18:30:49.460Z",
  "input": {
    "source": {
      "fileId": "5qTYa3gzN9gYUb5SzqUhqg"
    },
    "rules": [
      {
        "find": {
          "type": "regex",
          "pattern": "[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"
        },
        "redactWith": {
          "type": "RectangleRedaction",
          "reason": "Redacted"
        }
      }
    ]
  },
  "state": "processing",
  "percentComplete": 0
}
```

Step 4: Check Status of the RedactionCreator Resource

- The process to generate **markup XML** runs asynchronously on the PrizmDoc server. The **POST request** you sent in Step 3 will return immediately and before the output is ready. This means you will need to check the status of the process by sending a **GET request** to the resource you just created.
- In response to this request, JSON will be returned that includes a `state` property. When this property is `complete`, the JSON response will also include an `output` property, which means you can proceed to the next step.
- See the [Redaction Creator API](#) for more details about this request.

Example

```
GET /v2/redactionCreators/Rr64ma-U_HseoPrs6y0iiw

200 OK
Content-Type: application/json
{
  "processId": "Rr64ma-U_HseoPrs6y0iiw",
  "expirationDateTime": "2014-12-03T18:30:49.460Z",
  "input": {
    "source": {
      "fileId": "5qTYa3gzN9gYUb5SzqUhqg"
    },
    "rules": [
      {
        "find": {
          "type": "regex",
          "pattern": "[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"
        },
        "redactWith": {
```

```
        "type": "RectangleRedaction",
        "reason": "Redacted"
      }
    ]
  },
  "state": "complete",
  "percentComplete": 100,
  "output": {
    "markupFileId": "01bLJwFGxf9QGUTkyrOqig"
  }
}
```

Step 5: Start the Markup Burning Process (Redaction)

- Using the **file IDs** you obtained for the source document in Step 1 and the **XML markup file** in Step 4, you can now start the process to redact the document by sending a **POST request**, which will start a process that runs asynchronously on the PrizmDoc server to produce a redacted document.

Example

```
POST /PCCIS/V1/MarkupBurner
Content-Type: application/json
{
  "input": {
    "documentFileId": "5qTYa3gzN9gYUb5SzqUhqg",
    "markupFileId": "01bLJwFGxf9QGUTkyrOqig"
  }
}

200 OK
Content-Type: application/json
{
  "processId": "bQpcuixhvGmNqn5ElskO6Q",
  "expirationDateTime": "2014-12-03T18:30:49.460Z",
  "input": {
    "documentFileId": "5qTYa3gzN9gYUb5SzqUhqg",
    "markupFileId": "01bLJwFGxf9QGUTkyrOqig"
  },
  "state": "processing",
  "percentComplete": 0
}
```

Step 6: Check Status of the MarkupBurner Resource

- The process to generate a redacted document runs asynchronously on the PrizmDoc server. The **POST request** you sent in Step 5 will return immediately and before the output is ready. This means you will need to check the status of the process by sending a **GET request** to the resource you just created.
- In response to this request, JSON will be returned that includes a `state` property. When this property is `complete`, the JSON response will also include an `output` property, which means you can proceed to the next step.
- See the [Markup Burner API](#) for more details about this request.

Example

```
GET /PCCIS/V1/MarkupBurner/bQpcuixhvGmNqn5ElskO6Q

200 OK
Content-Type: application/json
{
  "processId": "bQpcuixhvGmNqn5ElskO6Q",
  "expirationDateTime": "2014-12-03T18:30:49.460Z",
  "input": {
    "documentFileId": "5qTYa3gzN9gYUb5SzqUhqg",
    "markupFileId": "o1bLJwFGxf9QGuTkyrOqig"
  },
  "state": "complete",
  "percentComplete": 100,
  "output": {
    "documentFileId": "5ufb3ytUb1BxxgSUAk_G9Q"
  }
}
```

Step 7: Download the Redacted Document

- Once the markup burning process completes successfully, the new, redacted PDF document is available for download.

Example

```
GET http://192.168.0.1:18681/PCCIS/V1/WorkFile/5ufb3ytUb1BxxgSUAk_G9Q

200 OK
Content-Type: application/pdf
[binary data]
```

Set up a Viewing Session for a CAD Drawing which has XREF Dependencies

How to Set Up a Viewing Session for a CAD Drawing Which Has XREF Dependencies

This topic explains how to set up a viewing session for a CAD drawing which depends upon external files via XREF.

Many documents, such as a PDF, are self-contained in a single file. Setting up a viewing session for these sorts of documents is relatively simple: after creating a viewing session, you make one HTTP call to upload the source document's file bytes.

Setting up a viewing session for a CAD drawing which is made up of multiple files is slightly more complicated. You will need to:

1. POST to create a new viewing session.

2. POST to upload each file to the work file API, both the primary CAD drawing and all its dependencies, creating distinct `fileId` values for each one.
3. POST a JSON object to the work file API to create a special "package" work file for the entire set. This "package" work file defines which of the files is the primary file for viewing, specifies path information for each of the files (typically relative to the primary file), and has its own distinct `fileId`.
4. Assign the "package" `fileId` to the viewing session so that the entire CAD drawing may be viewed.

This topic walks through examples of this in detail. It has two parts:

- [Preparing to view a CAD XREF document for the first time](#)
- [Preparing to view a CAD XREF document which has already been uploaded](#)

Preparing to view a CAD XREF document for the first time

Step 1: Identify the necessary files and their local paths

Imagine a CAD engineer has prepared a master.dwg file which depends on several other files and has stored all of the files on the disk in the following way:

```
./master.dwg
./parts/desk.dwg
./parts/desk-extension.dwg
./other/chair.dwg
./common/logo.png
```

In order for other CAD engineers to be able to open master.dwg and view the entire contents, they will need all of these files, stored in the same way as they are here.

In the same way, in order for PrizmDoc to prepare master.dwg for viewing by an end user in the browser, it too will need all of these files and information about how they were stored by the CAD engineer on the local disk.

Step 2: Create a new viewing session

Submit a POST to creating a new viewing session:

```
POST /PCCIS/V1/ViewingSession
Content-Type: application/json

{
  "render": {
    "html5": {
      "alwaysUseRaster": false
    }
  }
}

HTTP/1.1 200
Content-Type: application/json

{
  "viewingSessionId": "AqQp3wrrrSZ5YhJoFdj44Z_rT4629XnO6j7bEjjSRA5fiQUljuiYgi-
ng9sP4v95VTVqilte6bsaC6eGpUulMUWid1VN9qwh6rN5wp82eSfM",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

Hold on to the `viewingSessionId` (and `affinityToken`, if provided). You will need it again later.

Step 3: Deliver the configured HTML5 viewer resources to the end user

Now that you have a `viewingSessionId`, you can go ahead and deliver the Viewer, configured with the given `viewingSessionId`, to your end user's browser. This allows their browser to start parsing the Viewer's JavaScript resources as early as possible.

NOTE: We currently do not support the Viewer's download option when viewing CAD with XREF. When configuring the Viewer, make sure you set `uiElements.download` to `false`.

Step 4: Upload the files to the back end

POST each of the necessary files to the work file API, creating a unique `fileId` for each one. **If you are using PrizmDoc Self-Hosted (in cluster-mode) or PrizmDoc Cloud, be sure to include the `affinityToken` value returned after creating the viewing session in an `Accusoft-Affinity-Token` header of each request.**

master.dwg:

```
POST /PCCIS/V1/WorkFile
Content-Type: application/octet-stream
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

<<master.dwg bytes>>

HTTP/1.1 200 OK
Content-Type: application/json

{
  "fileId": "CVBuD7DbQYNoJDqByGierQ",
  "fileExtension": "dwg",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

parts/desk.dwg:

```
POST /PCCIS/V1/WorkFile
Content-Type: application/octet-stream
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

<<parts/desk.dwg bytes>>

HTTP/1.1 200 OK
Content-Type: application/json

{
  "fileId": "5qTYa3gzN9gYUb5SzqUhgq",
  "fileExtension": "dwg",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

parts/desk-extension.dwg:

```
POST /PCCIS/V1/WorkFile
```

```
Content-Type: application/octet-stream
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

<<parts/desk-extension.dwg bytes>>

HTTP/1.1 200 OK
Content-Type: application/json

{
  "fileId": "o1bLJwFGxf9QGuTkyrOqig",
  "fileExtension": "dwg",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

other/chair.dwg:

```
POST /PCCIS/V1/WorkFile
Content-Type: application/octet-stream
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

<<other/chair.dwg bytes>

HTTP/1.1 200 OK
Content-Type: application/json

{
  "fileId": "KxonBTYJyRpCswOLn_paiw",
  "fileExtension": "dwg",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

common/logo.png:

```
POST /PCCIS/V1/WorkFile
Content-Type: application/octet-stream
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

<<common/logo.png bytes>>

HTTP/1.1 200 OK
Content-Type: application/json

{
  "fileId": "JcskB6w8D5_qlf40A3xspQ",
  "fileExtension": "png",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

At this point, you have a unique `fileId` for each of the uploaded files.

Step 5: Create a final "package" work file

This is the step where you tie all of the files together under a single new `fileId` and provide some crucial information which is necessary for the back end to actually prepare the content for viewing.

When uploading simple work files, the POST body is simply the bytes of the file. However, by submitting a POST with a `Content-Type` of `application/json`, you can instead instruct the back end to create a new "package" work file from a set of existing work files.

A "package" file defines the following:

- A list of files in the package, specified by `fileId`.
- A unique path for each file in the package.
- The primary file in the package, identified by its path.

To create a new "package" for the files already uploaded in Step 4 above, we would submit a POST like this:

```
POST /PCCIS/V1/WorkFile
Content-Type: application/json
Accusoft-Affinity-Token: ejN9/kXEYOUken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

{
  "primaryPath": "master.dwg",
  "items": [
    { "fileId": "CVBuD7DbQYNoJDqByGierQ", "path": "master.dwg" },
    { "fileId": "5qTYa3gzN9gYUb5SzqUhqg", "path": "parts/desk.dwg" },
    { "fileId": "01bLJwFGxf9QGUTkyrOqig", "path": "parts/desk-extension.dwg" }
  ],
  { "fileId": "KxonBTYJyRpCswOLn_paiw", "path": "other/chair.dwg" },
  { "fileId": "JcskB6w8D5_qlf40A3xspQ", "path": "common/logo.png" }
]
}

HTTP/1.1 200 OK
Content-Type: application/json

{
  "fileId": "nkG9fiAmj27X3MhqGdbsXA",
  "fileExtension": "dwg",
  "primaryPath": "master.dwg",
  "items": [
    { "fileId": "CVBuD7DbQYNoJDqByGierQ", "path": "master.dwg" },
    { "fileId": "5qTYa3gzN9gYUb5SzqUhqg", "path": "parts/desk.dwg" },
    { "fileId": "01bLJwFGxf9QGUTkyrOqig", "path": "parts/desk-extension.dwg" }
  ],
  { "fileId": "KxonBTYJyRpCswOLn_paiw", "path": "other/chair.dwg" },
  { "fileId": "JcskB6w8D5_qlf40A3xspQ", "path": "common/logo.png" }
],
  "affinityToken": "ejN9/kXEYOUken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

The new `fileId` which is returned ("nkG9fiAmj27X3MhqGdbsXA") is a single id we can use to refer to this entire set of files, with `master.dwg` being the primary file for viewing.

Step 6: Attach the "package" work file to the viewing session

Now that your "package" work file exists, you can attach it as the source document for the viewing session with this PUT request. > **NOTE:** The `viewingSessionId` used in the URL is prefixed with the letter `u:**`

```
PUT
/PCCIS/V1/ViewingSession/uAqQp3wrrSZ5YhJoFdj44Z_rT4629XnO6j7bEjjSRA5fiQUljuiYgi-
ng9sP4v95VTVqilte6bsaC6eGpUulMUwid1VN9qwh6rN5wp82eSfM/SourceRef
```

```
Content-Type: application/json

{
  "refType": "workFile",
  "fileId": "nkG9fiAmj27X3MhqGdbsXA"
}

HTTP/1.1 200 OK
```

(If you are wondering, an `Accusoft-Affinity-Token` is not required in this particular request; the viewing session id itself serves the role of the affinity token.)

At this point, the back end will begin converting the CAD content for viewing in the browser and delivering content to the end user as it becomes ready.

Preparing to view a CAD XREF document which has already been uploaded

All work files do eventually expire, and there is never a guarantee that a particular `fileId` will be available. However, each time you use a `fileId` as the source document of a viewing session, we will extend the amount of time that `fileId` will remain available. In many cases, you can simply continue reusing the existing `fileId` when creating a new viewing session for the same CAD drawing.

Step 1: Create a new viewing session

Submit a POST to create a new viewing session. **If you are using PrizmDoc Self-Hosted (in cluster-mode) or PrizmDoc Cloud, make sure that you add an `Accusoft-Affinity-Token` header which matches the `affinityToken` of the work file. This is crucial to ensure that the viewing session will be created on a machine in the cluster that will actually have access to the existing `fileId` in the next step.**

```
POST /PCCIS/V1/ViewingSession
Content-Type: application/json
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

{
  "render": {
    "html5": {
      "alwaysUseRaster": false
    }
  }
}

HTTP/1.1 200
Content-Type: application/json

{
  "viewingSessionId": "HO9MagQSyIizShhzLpjp-H73-
IRtoqcPlJZmLP0PPwI3HxJ36Ds_HunbqMmtKfTlgt4h4-
96X67t7onW2P9XFV5Rw4pSddBrNoFp4A8bdFw",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

Step 2: Attach the existing "package" work file to the viewing session

Simply attach the existing `fileId` to the new viewing session as you did before. **Remember that the `viewingSessionId` used in this URL must be prefixed with the letter `u`:**

```
PUT /PCCIS/V1/ViewingSession/uH09MagQSyIizShhzLpjp-H73-
IRtoqcPlJZmLP0PPwI3HxJ36Ds_HunbqMmtKfT1gt4h4-
96X67t7onW2P9XfV5Rw4pSddBrNoFp4A8bdFw/SourceRef
Content-Type: application/json

{
  "refType": "workFile",
  "fileId": "nkG9fiAmj27X3MhqGdbsXA"
}

HTTP/1.1 200 OK
```

(Although it was crucial you provided an `Accusoft-Affinity-Token` in the previous POST request, it is not required in this PUT request; the viewing session id itself serves the role of the affinity token.)

That's it. The new viewing session is ready to provide viewing content for the previously-uploaded CAD drawing files.

What if the existing "package" work file has expired?

When you make the PUT request to associate the existing `fileId` to the new viewing session, the PUT request will fail if no such `fileId` can be found:

```
HTTP/1.1 480 NotFound
Content-Type: application/json

{
  "errorCode": "NotFound",
  "errorDetails": {
    "in": "body",
    "at": "fileId"
  }
}
```

If the `fileId` has indeed expired, this is the response you will receive. At this point, you will need to re-upload all of the CAD files again and create a new "package" `fileId` which you can then attach to the viewing session.

Note that this error technically only means that the specified `fileId` could not be found, and it may occur for several reasons:

- The `fileId` has expired.
- The `fileId` value itself was incorrect.
- You forgot to provide the correct `Accusoft-Affinity-Token` header value when creating the viewing session, and as a result the viewing session has been assigned to a machine in the cluster that does not have access to the work file (even though the work file itself may still exist). To prevent this, make sure you are providing the work file's `affinityToken` value in an `Accusoft-Affinity-Token` header when making the POST request to create the new viewing session.

Use the Markup JSON Schema

Introduction

The [PrizmDoc Server](#) and [PAS Markup Burner APIs](#) support JSON Markup content and use a JSON Schema definition to validate incoming JSON content for structure, required properties, and value types. You can validate JSON Markup against this Schema manually to determine root causes of schema validation errors.

See the [Markup JSON Specification](#) topic for a human-readable version of the schema.

Usage

You can use one of the libraries/tools listed in the [The JSON Schema documentation](#) to validate your Markup content outside of PrizmDoc using your language of choice. PrizmDoc uses [everit-org/json-schema](#) internally.

Caveats

Because the Schema definition uses a `oneOf` directive to compare against all of our possible mark types, most validators will return error messages for *all mark types*, not just the type you specify in the markup type property. You will need to filter for the errors related to the Schema that most closely matches the mark type.

Download the Schema

[Click here](#) to view the JSON Schema definition. To download it, right-click on the same link and select "Save link/target as...".

Use a Viewing Session

Introduction

This section provides instructions on how to use a Viewing Session in PrizmDoc Server:

- [Step 1: Create a Viewing Session](#)
- [Step 2: \(Optional\) Upload the Source Document](#)
- [Step 3: \(Optional\) Start the Session](#)
- [Step 4: Request Content](#)
- [Step 5: \(Optional\) Stop the Session](#)

Step 1: Create a Viewing Session

The recommended method of creating a Viewing Session is through the [POST /ViewingSession endpoint in PAS](#).

The body of the request specifies details of how the Viewing Session's content will be generated and how its source document will be transferred to PrizmDoc, but does not necessarily provide the source document itself.

Example Request

```
http://localhost:18681/PCCIS/V1/ViewingSession

Content-Type: application/json
{
  "render": {
```



```
    "html5": {
      "alwaysUseRaster": false
    }
  }
}
```

Example Response

```
200 OK

Content-Type: application/json
{
  "viewingSessionId": "-gchUEYvBE5OCgcWJajgoXcW7QD0I8zNDFlexD9hzbXkmrYlw8DrxJ-
KiHAf2oTAL_HiHK1MsstBlNgZFCrcJQ"
}
```

Step 2: (Optional) Upload the Source Document

If a Viewing Session was created with a `documentSource` of "api", a second request is required to provide it with the source document. For other source types, the source document will have already been specified in the `externalId` property, and this step should be skipped.

Example Request

```
http://localhost:18681/PCCIS/V1/ViewingSession/u-
gchUEYvBE5OCgcWJajgoXcW7QD0I8zNDFlexD9hzbXkmrYlw8DrxJ-
KiHAf2oTAL_HiHK1MsstBlNgZFCrcJQ/SourceFile

Content-Type: application/octet-stream

{binary data}
```

Example Response

```
200 OK
```

Step 3: (Optional) Start the Session

By default, content generation for a Viewing Session does not begin until content is first requested from it. However, it is possible to manually kick off conversion beforehand in order to improve the response time of subsequent content requests.

Example Request

```
http://localhost:18681/PCCIS/V1/ViewingSession/u-
gchUEYvBE5OCgcWJajgoXcW7QD0I8zNDFlexD9hzbXkmrYlw8DrxJ-
```

```
KiHAF2oTAL_HiHK1MsstBlNgZFCrcJQ/Notification/SessionStarted  
  
Content-Type: application/json  
{  
  "Viewer": "HTML5"  
}
```

Example Response

```
200 OK
```

Step 4: Request Content

Once a Viewing Session has been created and provided with its source document, it becomes possible to make requests for content from it, as described in the [HTML5 Viewing API](#) reference.

Example Request

```
http://localhost:18681/PCCIS/V1/Page/q/0?DocumentID=u-  
gchUEYvBE5OCgcWJajgoXcW7QD0I8zNDFlexD9hzbXkmrYlw8DrxJ-  
KiHAF2oTAL_HiHK1MsstBlNgZFCrcJQ&ContentType=png
```

Example Response

```
200 OK  
  
Content-Type: image/png  
  
{image data}
```

Step 5: (Optional) Stop the Session

A Viewing Session may be made unavailable prior to the time it would normally expire with a `SessionStopped` request. See the [API reference](#) for details of what properties can be set for the error response returned for subsequent requests against the stopped session.

Example Request

```
http://localhost:18681/PCCIS/V1/ViewingSession/u-  
gchUEYvBE5OCgcWJajgoXcW7QD0I8zNDFlexD9hzbXkmrYlw8DrxJ-  
KiHAF2oTAL_HiHK1MsstBlNgZFCrcJQ/Notification/SessionStopped
```

Example Response

200 OK

Watermark Content in a Viewing Session

Introduction

The PrizmDoc Services supports Text, Diagonal Text and Image watermarks for a viewing session. This means that the viewable content that is displayed in the browser will contain the specified watermarks. The watermarks will not be applied to the source document.

NOTE: Specifying watermarks in a viewing session will disable cache reuse. This means that the process to convert a document to viewable content will be executed for each viewing session even if the source document is the same as one in an existing viewing session. See the topic [Adjusting Caching Parameters for PrizmDoc Server](#) for more information about cache reuse and the side-effects of disabling it.

NOTE: Watermarks are currently only supported when viewing some SVG content. When viewing CAD files such as .dwg, .dxf, .dwt or .dgn, then watermarks are not supported.

NOTE: If viewing watermarked raster content, then multi-line text, text decoration, and complex Unicode fonts (for example, Hebrew, Arabic) are not supported.

Applying Watermarks

Watermarks are defined by setting additional JSON properties in the body of the HTTP request that you send to create a new Viewing Session. These properties are described in more detail in the [Viewing Sessions](#) topic under the **POST ViewingSession** request.

The watermark properties you set for the viewing session will be applied to all pages of the document. You can specify more than one watermark in a viewing session. You can also apply watermarks of mixed types. For example, a diagonal text watermark can be used to apply the text "Confidential" across each page and an image watermark can be added to apply your company logo to the top-right corner of each page.

Text Watermarks

The following sections describe special characteristics of text watermarks that warrant additional explanation.

Special Characters

The control character `\n` has a special meaning inside the text string of a text watermark. If present, a line break will be applied at its position.

Example

```
"text": "ACME Corporation\nConfidential"
```

Dynamic Page Number and Page Count

Use the special replacement syntax `"{{pageNumber}}"` and `"{{pageCount}}"` in the text string of a text watermark to

display the current page number and/or page count on each page.

Example

```
"text": "Page {{pageNumber}} of {{pageCount}}"
```

Font Considerations

The "fontFamily" property can be used to specify a font for text watermarks.

When viewing SVG content, please be aware that text watermarks will be created using SVG that is rendered on the browser. This means that the font specified in this property must be available on the Viewer. When the "fontFamily" property is not set, the default font of the client browser executing the Viewer will be used to render text watermarks.

Text Length Considerations

The PrizmDoc Server API does not define a limit for the watermark text length. Note that very long text watermarks may not fit entirely onto the document page and will not be fully visible.

Image Watermarks

The following sections describe special characteristics of image watermarks that warrant additional explanation.

Supported Formats

The source format of image watermarks must be PNG. Transparency options of a PNG will be honored so that transparent sections of the image will reveal the page content beneath it.

The PrizmDoc Server API does not define a limit for the watermark image size or resolution.

Source Locations

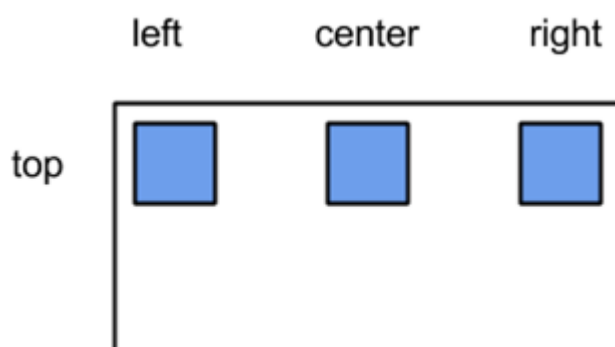
The source of image watermarks can be either a URL or work file ID.

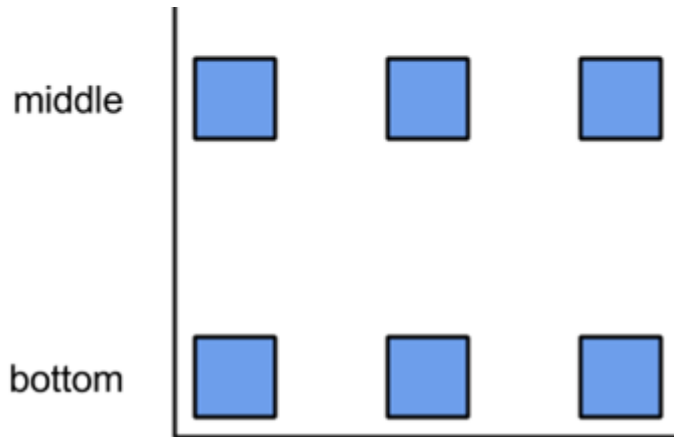
The URL should specify an absolute location with a scheme of HTTP or HTTPS. Please be sure that the URL is accessible from the server where PrizmDoc Services are running.

See the [Work Files](#) topic for more information about creating work files and getting their IDs.

Aligning Watermarks

Text and Image watermarks can be aligned with nine commonly used locations in a page. Diagonal Text can only be aligned in the center of the page.





You can select any one of these nine positions for Text and Image watermarks using two properties:

- **horizontalAlign** - May be "left", "right", or "center". Default is "center".
- **verticalAlign** - May be "bottom", "middle", or "top". Default is "middle".

For example, to place a text watermark at the bottom-right corner of a page set **horizontalAlign** to **right** and **verticalAlign** to **bottom**.

Setting the **autoSize** property for an Image watermark will override these alignment settings.

Examples

The following examples demonstrate how to apply the various types of watermarks.

Text Watermark Example

This example demonstrates how to apply a single text watermark. In the sample output, notice that the `\n` control character in the text string creates line breaks in the text.

Example

```
POST http://localhost:18681/PCCIS/V1/ViewingSession Content-Type:
application/json
{
  "tenantId": "my application name",
  "externalId": "my-unique-document-name.docx",
  "render": {
    "html5": {
      "alwaysUseRaster": false
    }
  },
  "watermarks": [
    {
      "type": "text",
      "opacity": 0.6,
      "text": "jdoe\n67.79.169.114\n11/13/2014 2:24 PM\nNOT FOR
DISTRIBUTION",
      "color": "red",
      "fontFamily": "Consolas",
      "fontSize": "16pt",
      "fontWeight": "bold",
      "verticalAlign": "bottom",
      "horizontalAlign": "right"
    }
  ]
}
```

Sample Output

The screenshot shows the PrizmDoc Viewer interface with a document titled "Biweekly Time Sheet". The document contains the following information:

[Company Name]

[Street Address] Pay period start date: 11/1/2013
[Address 2] Pay period end date: 11/14/2013
[City, ST ZIP Code]

Employee: Bill Bilson Employee SSN: 123-31-5645
Manager: Joe Smith Employee e-mail: employee@company.com

Day		Regular Hours	Overtime Hours	Sick	Vacation	Total
Monday	11/1/2013					
Tuesday	11/2/2013					
Wednesday	11/3/2013					
Thursday	11/4/2013					
Friday	11/5/2013					
Saturday	11/6/2013					
Sunday	11/7/2013					
Monday	11/8/2013					
Tuesday	11/9/2013					
Wednesday	11/10/2013					
Thursday	11/11/2013					
Friday	11/12/2013					
Saturday	11/13/2013					
Sunday	11/14/2013					
Total hours						
Rate per hour						
Total pay						

Employee signature _____ Date _____
Manager signature _____ Date _____

jd
11/13/2014 2:24 PM
NOT FOR DISTRIBUTION

Diagonal Text Watermark Example

This example demonstrates how to apply a single diagonal text watermark. Diagonal text watermarks can only be applied to the center of the page. In the sample output, the text "Accusoft\nConfidential" is displayed diagonally over the center of the page. Notice that the \n control character in the text string creates a line break in the text.

Example

```

POST http://localhost:18681/PCCIS/V1/ViewingSession Content-Type:
application/json
{
  "tenantId": "my application name",
  "externalId": "my-unique-document-name.docx",
  "render": {
    "html5": {
      "alwaysUseRaster": false
    }
  },
  "watermarks": [
    {
      "type": "diagonalText",
      "slope" : "up",
      "opacity": 0.25,
      "text": "Accusoft\nConfidential",
      "fontSize": "50pt",
      "color": "#FF0000",
      "fontWeight": "bold",
      "fontFamily": "Arial"
    }
  ]
}

```

Sample Output

Biweekly Time Sheet

[Company Name]

[Street Address] Pay period start date: 11/1/2013
 [Address 2] Pay period end date: 11/14/2013
 [City, ST ZIP Code]

Employee: Bill Bilson Employee SSN: 123-31-5645
 Manager: Joe Smith Employee e-mail: employee@company.com

Day		Regular Hours	Overtime Hours	Sick	Vacation	Total
Monday	11/1/2013					
Tuesday	11/2/2013					
Wednesday	11/3/2013					
Thursday	11/4/2013					
Friday	11/5/2013					
Saturday	11/6/2013					
Sunday	11/7/2013					
Monday	11/8/2013					
Tuesday	11/9/2013					
Wednesday	11/10/2013					
Thursday	11/11/2013					
Friday	11/12/2013					
Saturday	11/13/2013					

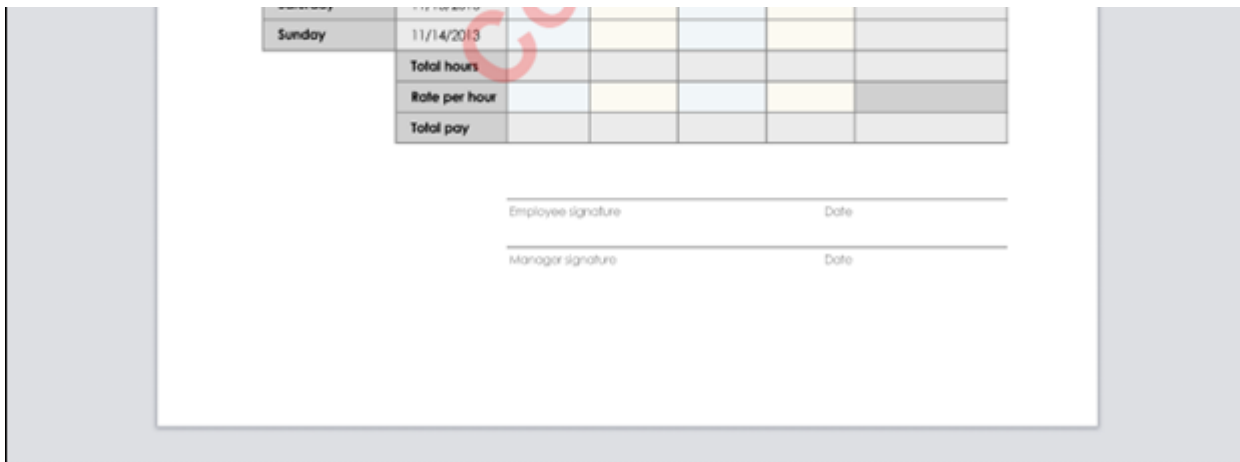


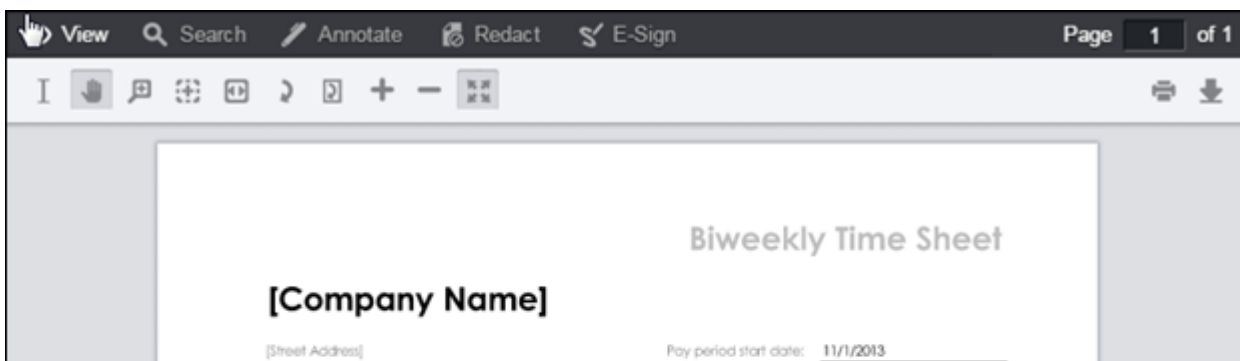
Image Watermark Example

This example demonstrates how to apply a single image watermark over the entire page. In the sample output, the tri-color Accusoft logo spans the center of the page. Notice that the transparent areas of the PNG image source do not obstruct the page content beneath it.

Example

```
POST http://localhost:18681/PCCIS/V1/ViewingSession Content-Type:
application/json
{
  "tenantId": "my application name",
  "externalId": "my-unique-document-name.docx",
  "render": {
    "html5": {
      "alwaysUseRaster": false
    }
  },
  "watermarks": [
    {
      "type": "image",
      "opacity": 0.3,
      "src": "http://localhost/watermark_images/logo.png",
      "autoSize": "fit"
    }
  ]
}
```

Sample Output



[Address 2] Pay period end date: 11/14/2013
 [City, ST ZIP Code]

Employee: Bill Bilson Employee SSN: 123-31-5645
 Manager: Joe Smith Employee e-mail: employee@company.com

Day		Regular Hours	Overtime Hours	Sick	Vacation	Total
Monday	11/1/2013					
Tuesday	11/2/2013					
Wednesday	11/3/2013					
Thursday	11/4/2013					
Friday	11/5/2013					
Saturday	11/6/2013					
Sunday	11/7/2013					
Monday	11/8/2013					
Tuesday	11/9/2013					
Wednesday	11/10/2013					
Thursday	11/11/2013					
Friday	11/12/2013					
Saturday	11/13/2013					
Sunday	11/14/2013					
Total hours						
Rate per hour						
Total pay						

 Employee signature Date

 Manager signature Date

Multiple Watermarks Example

This example demonstrates how to apply multiple watermarks to a viewing session. In the sample output, the tri-color Accusoft logo spans the center of the page and the text watermark near the bottom-right corner displays information about the user.

Example

```
POST http://localhost:18681/PCCIS/V1/ViewingSession Content-Type:
application/json
{
  "tenantId": "my application name",
  "externalId": "my-unique-document-name.docx",
  "render": {
    "html5": {
      "alwaysUseRaster": false
    }
  },
  "watermarks": [
    {
      "type": "text",
      "opacity": 0.6,
      "text": "jdoe\n67.79.169.114\n11/13/2014 2:24 PM\nNOT FOR
DISTRIBUTION",
      "color": "red",
```

```
    "fontFamily": "Consolas",
    "fontSize": "16pt",
    "fontWeight": "bold",
    "verticalAlign": "bottom",
    "horizontalAlign": "right"
  },
  {
    "type": "image",
    "opacity": 0.3,
    "src": "http://localhost/watermark_images/logo.png",
    "autoSize": "fit"
  }
]
```

Sample Output

Biweekly Time Sheet

[Company Name]

[Street Address] Pay period start date: 11/1/2013
[Address 2] Pay period end date: 11/14/2013
[City, ST ZIP Code]

Employee: Bill Bilson Employee SSN: 123-31-5645
Manager: Joe Smith Employee e-mail: employee@company.com

Day		Regular Hours	Overtime Hours	Sick	Vacation	Total
Monday	11/1/2013					
Tuesday	11/2/2013					
Wednesday	11/3/2013					
Thursday	11/4/2013					
Friday	11/5/2013					
Saturday	11/6/2013					
Sunday	11/7/2013					
Monday	11/8/2013					
Tuesday	11/9/2013					
Wednesday	11/10/2013					
Thursday	11/11/2013					
Friday	11/12/2013					
Saturday	11/13/2013					
Sunday	11/14/2013					
Total hours						
Rate per hour						
Total pay						

Employee signature _____ Date _____
Manager signature _____ Date _____

jdoe
11/13/2014 2:24 PM

NOT FOR DISTRIBUTION

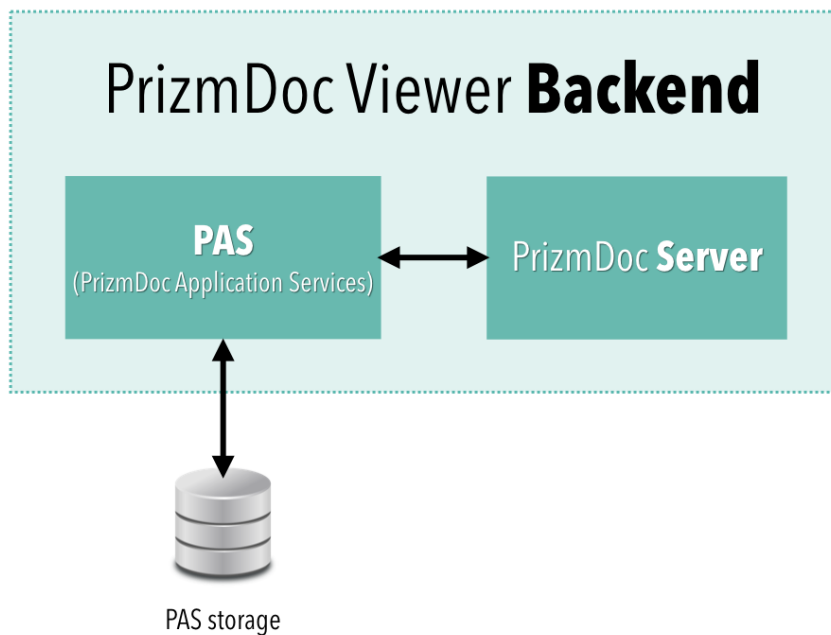
Administrator Guide

Administrator Guide (Self-Hosted)

This section explains how to deploy and administer your own self-hosted PrizmDoc Viewer backend.

Architecture Overview

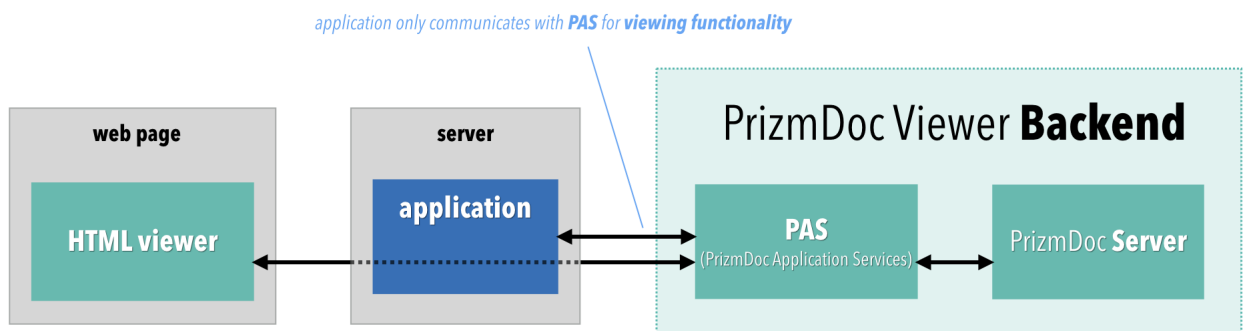
The PrizmDoc Viewer backend is comprised of two tiers, PAS and PrizmDoc Server:



- **PrizmDoc Server**, the technical heart of the product, is a document processing and conversion engine. It is compute intensive but has no permanent storage.
- **PAS** is a layer in front of PrizmDoc Server responsible for viewing concerns, such as saving and loading of annotations. As such, PAS requires access to a storage system which you manage (there are several supported options; see more below).

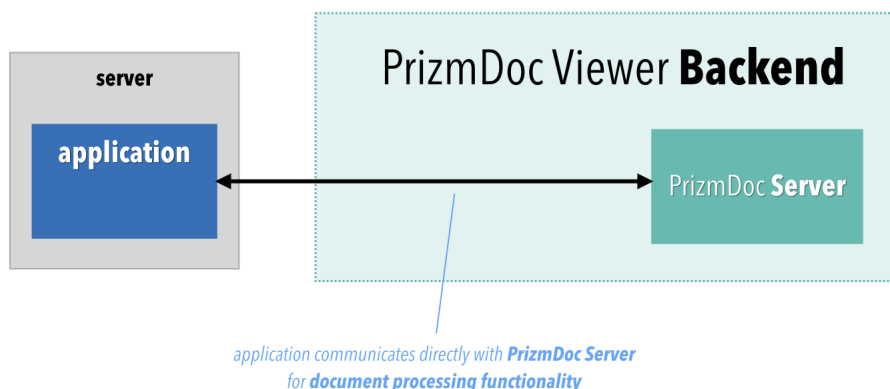
An application can send REST API requests to *both* of these tiers:

- For **viewing**, an application will send HTTP requests to PAS (and PAS will, in turn, send requests to PrizmDoc Server):



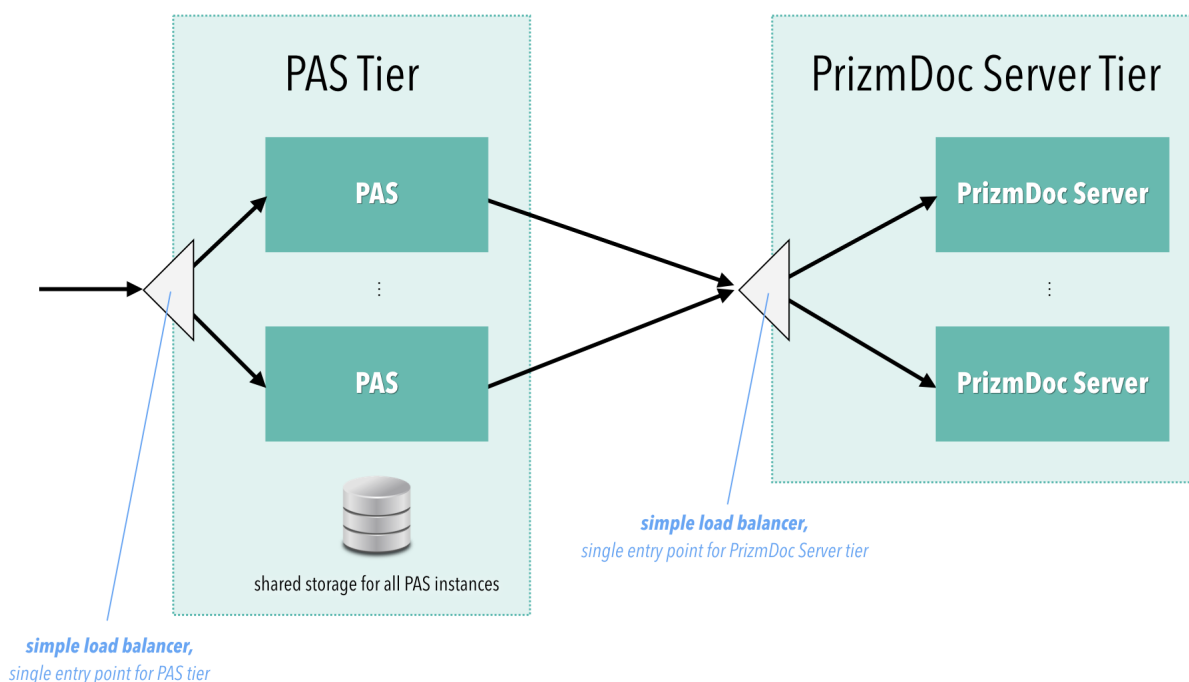
- For **document processing** (like file conversion or redaction), an application will send HTTP requests directly to

PrizmDoc Server:



Load Balancing and Cluster Management

It is common in a production deployment to run multiple instances of PAS and PrizmDoc Server. The PAS and PrizmDoc Server tiers are conceptually independent, and each should be fronted by a load balancer of your choice:



Application developers needing to make PAS or PrizmDoc Server REST API calls should only send their requests to the load balancer sitting in front of the tier they are calling.

For PAS, any incoming request to the PAS tier can be routed to any PAS instance. You can use any off-the-shelf load balancer in front of your PAS instances (there is no need for anything like sticky sessions). Any PAS instance is capable of directly handling any request.

For PrizmDoc Server, the process is similar. Any incoming request to the PrizmDoc Server tier can be sent to any PrizmDoc Server instance. You can use any off-the-shelf load balancer in front of your PrizmDoc Server instances (there is no need for anything like sticky sessions). However, this is not because any PrizmDoc Server is capable of *directly* handling a request. Rather, this is because PrizmDoc Server instances are capable of routing any incoming request to the correct PrizmDoc Server instance in the cluster (the instance where document processing is actually occurring).

In order for PrizmDoc Server's automatic routing to work correctly, each PrizmDoc Server instance MUST have an up-to-date, accurate list of all of the instances in the PrizmDoc Server cluster. Each time you add or remove PrizmDoc Server instances to

your cluster, you must inform each instance that the server list has changed (you can do this with a REST API call to each instance). For more information about this, see [Cluster Server Environments > PrizmDoc Server](#).

Supported Storage Options for PAS

Your PAS instances will need to be configured to use some sort of shared storage. PAS supports several options for this:

- File system (such as network attached storage)
- Microsoft SQL Server
- MySQL
- Amazon S3
- Azure Blob Storage (beta)

For more information, see [PAS Configuration](#).

Licensing

Self-hosted licensing is based upon your use of PrizmDoc Server. Each PrizmDoc Server instance you run must be configured with your license key (PAS instances do not require a license key to run).

For your convenience, PrizmDoc Server automatically runs in evaluation mode without a license: images may be displayed with a watermark on them and occasionally dialogs may be posted reminding you that PrizmDoc Viewer is in evaluation mode with a fixed feature set. When you are ready to deploy to production, a license will be required to run PrizmDoc Server in your production environment. Please contact info@accusoft.com to obtain a deployment license. For more information, see [Deployment Licensing](#).

Deployment Options

For both PAS and PrizmDoc Server, we offer ready-to-run Docker images as well as Windows and Linux installers.

PAS

For PAS, deploying is easiest with Docker, and we recommend you use the PAS Docker image if possible.

	PAS Pre-Installed
Docker	Yes
Windows Installer	No
Linux Installer	No

To use the PAS Docker image, see [Install PAS > Using Docker](#).

To install PAS directly on Windows, see [Install PAS > Install on Windows](#).

To install PAS directly on Linux, see [Install PAS > Install on Linux](#).

PrizmDoc Server

For PrizmDoc Server, deploying is easiest with Docker, and we recommend you use the PrizmDoc Server Docker image if possible.

It largely depends on whether you need to render with Microsoft Office. If you do, then your only option is to deploy to a Windows machine with our Windows installer. However, if you plan to use PrizmDoc Server's built-in LibreOffice rendering, you can use any of our deployment options, including the Docker image.

	PrizmDoc Server Pre-Installed	Office Rendering
Docker	Yes	LibreOffice

	PrizmDoc Server Pre-Installed	Office Rendering
Windows Installer	No	LibreOffice or Microsoft Office
Linux Installer	No	LibreOffice

To use the PrizmDoc Server Docker image, see [Install PrizmDoc Server > Using Docker](#).

To use the PrizmDoc Server Windows installer, see [Install PrizmDoc Server > Install on Windows](#).

To use the PrizmDoc Server Linux installer, see [Install PrizmDoc Server > Install on Linux](#).

Minimal Backend Quick Start

If you just want to quickly setup a single instance of PAS and PrizmDoc Server running on a single machine (perhaps for evaluation), check out our [Minimal Backend Quick Start](#).

PrizmDoc Server

This section contains the following information:

- [Server Sizing](#)
- [Security Guidance](#)
- [Document Rendering Specifics](#)
- [Installing](#)
 - [Using Docker](#)
 - [Install on Windows](#)
 - [Windows Requirements & Supported Environments](#)
 - [Install on Windows](#)
 - [Registry Changes](#)
 - [Natively Render MSO Documents](#)
 - [Unattended Install & Uninstall](#)
 - [Uninstall PrizmDoc Server on Windows](#)
 - [Installing with Traditional Linux Install Packages \(deprecated\)](#)
 - [Requirements & Supported Environments for Traditional Linux Install Packages](#)
 - [Install Using Traditional Linux Install Packages](#)
 - [Install Asian Fonts on Traditional Linux Install Packages](#)
 - [Install Using Traditional Linux Install Packages on a Headless Environment](#)
 - [Uninstall Traditional Linux Install Packages](#)
 - [Check PrizmDoc Server Health](#)
 - [Upgrade PrizmDoc Viewer](#)
 - [Configure a Cluster](#)
- [Licensing](#)
 - [Metered License](#)
 - [OEM License](#)
 - [Cloud License \(Deprecated\)](#)
 - [Node-Locked License \(Deprecated\)](#)
- [Configuring](#)
 - [Central Configuration](#)
 - [PCCIS Configuration](#)
 - [Implement Caching Strategies](#)
 - [Adjust Caching Parameters](#)
 - [Change Encryption Keys for Public use Token Generation](#)
 - [Configure Microsoft Office Conversion Connectivity](#)
 - [Substitute Fonts for Office Rendering Fidelity](#)
 - [Upgrade from Legacy Configuration](#)

- [Clustering](#)
 - [Optimize Cache Performance for Cluster Mode](#)
 - [Affinity Tokens & Cluster Mode](#)
- [Starting & Stopping](#)
 - [Linux](#)
 - [Windows](#)

Server Sizing

Overview

This topic provides some general guidance on the factors that will impact performance of PrizmDoc Server so that you can adequately plan your deployment to ensure that you have sufficient hardware to handle your needs. You should consider the type of documents that you will be processing and based on the information provided below, adjust our recommendations accordingly to fit your use-case. If you have more specific questions regarding your specific usage of PrizmDoc Server, please contact our [Support team](#).

- [How Content Can Affect These Recommendations](#)
- [How Hardware Can Affect System Performance](#)
- [Example Server Configurations](#)
- [Additional Considerations for Pre-Conversion Services](#)

How Content Can Affect These Recommendations

PrizmDoc Server can process many different types of files. The majority of files processed by PrizmDoc Server displayed in the Viewer are PDF, Microsoft Office formats (Word, PowerPoint, and Excel), and scanned images. The system is capable of processing many files types outside of this basic set. For benchmarking purposes, we focus on processing the more commonly used formats.

The content of files can vary greatly, and their construction can affect PrizmDoc Server's ability to process them. Any of the factors mentioned below could cause file processing and rendering time to deviate from the performance expected from a basic document set.

Document Elements

The number of elements that exist within a file will affect how quickly the PrizmDoc Server services can convert the content in the necessary viewing format. CAD Drawings and PDF Documents using path elements to represent CAD data will contain many elements and require more time to convert. Elements within a file are not limited to the path elements, but text content can also be a factor.

Image Size

Images can be content on their own, but images are also embedded into PDF and Office files. The size of the images that the system needs to process can affect the system's ability to load the content being supplied. As the size of the image(s) increases, the amount of time needed to process the content increases. For example, the PDF standard allows for images to be stored within the file, while only displaying a small portion of the image. While the image may appear to be small in size, the information stored in the file can be larger than expected.

File Size

Uploading a file to the server before it can be processed will take some time. The size of a file can also be representative of the complexity of the file. As the file size increases, the amount of time to transfer and load the data will increase.

File Types

Certain file types, by their nature, may contain more of the above defined factors. CAD drawings will contain many line and path elements to represent the information that is needed. As these files grow in size, they will contain more data and require more processing for the system to generate content for display.

How Hardware Can Affect System Performance

There is not one single piece of hardware that will address the performance concerns that you might have. In order to avoid under-utilizing your hardware, Accusoft recommends keeping these different characteristics in balance within your environment.

Disk IO

In its default configuration on common hardware of the day, the biggest limiting resource for PrizmDoc Server is most likely to be disk IO. PrizmDoc Server caches a considerable amount of data on a local drive consisting of document data, both original and converted, as well as state and other information.

As the load increases on a server, the data being written and read from disk will continue to grow with each new request. We often see customers proposing hardware configurations that possess high-end CPUs but only a single, ~7200 RPM Hard Disk Drive. A single IO device in this situation can quickly become overburdened, causing the queue length for the IO device to grow to the point where much of the actual time used for document conversions is spent waiting for IO requests. There are options today that can greatly improve IO transfer rates:

- SSD drives: A good option to minimize the IO wait time, allowing for the use of high-end multi-core processors to extract the most performance out of PrizmDoc Server.
- Shared Memory Drive (Linux): In this case, a chunk of the available RAM is shared and appears as a mounted file system. This storage is not persistent, but this is acceptable for the PrizmDoc Server cache with the understanding that PrizmDoc Server will reconvert documents to rebuild the cache if it is deleted.

Maximizing the number of Disk I/O operations your system can perform will be the most beneficial investment you can make in order to allow the system hosting PrizmDoc Server to perform at its peak.

CPU

The CPU is another resource that should be considered when determining hardware requirements for PrizmDoc Server. Assuming PrizmDoc Server performance is not bottlenecked by Disk IO (either SSD, Shared Memory, or other Disk IO optimization is in use), the ideal range for CPU usage is about 60% total, across all available logical cores. The 60% target is a guideline to where system activity should be for the system to be at peak productivity.

RAM

We recommend each server have 4GB of memory per logical core and no less than 16GB total memory. If you intend to use a RAM Disk or Shared Memory Volume, you'll need to account for that separately.

Example Server Configurations

The throughput expressed in this document is defining the number of unique documents the system can convert for viewing per minute. If a document has already been converted for display and is cached, the system resources needed to transfer that data are minimal and are not considered in these estimates. The documents included in the recommendation represent a standard mix of Office and PDF files ranging in size from a few pages to a few hundred pages.

Minimum Recommendation

- Windows: 5 per minute
- Linux: 10 per minute

Hardware Details

- Logical Cores: 4
- Memory: 16GB
- Hard Drive Type: SSD
- AWS: m5.xlarge

Moderate Recommendation

- Windows: 10 per minute
- Linux: 18 per minute

Hardware Details

- Logical Cores: 8
- Memory: 32GB
- Hard Drive Type: SSD
- AWS: m5.2xlarge

High-End Recommendation

- Windows: 15 per minute
- Linux: 30 per minute

Hardware Details

- Logical Cores: 16
- Memory: 64GB
- Hard Drive Type: SSD
- AWS: m5.4xlarge

Additional Considerations for Pre-Conversion Services

When enabling the [Work with Viewing Packages](#) feature, there are additional considerations for hardware requirements. Since the Viewing Package creation process creates all of the necessary artifacts that a future Viewing Session may require, it is much more computationally and resource expensive than an on-demand Viewing Session. As a general rule-of-thumb, creating a Viewing Package is about as resource intensive as having 5 concurrent users trying to view a document that has been uploaded for an on-demand Viewing Session.

When creating a Viewing Package, PrizmDoc will begin generating all of the artifacts for the document that is uploaded (e.g., svg, text, etc.). This is fundamentally different from on-demand Viewing Session creation in that PrizmDoc will only generate the artifacts that are immediately needed for viewing. Since everything is being converted at once, this puts additional strain on the PrizmDoc Back-end Services.

Security Guidance

Introduction

This topic covers the essential items that you should consider before deploying your application. For example, how [PrizmDoc Server](#) is designed, [ports](#) that need to be open for single-server and cluster modes, PrizmDoc Server

[administration](#), and creating [secure viewing sessions](#).

PrizmDoc Server

PrizmDoc Server is designed to run as an internal web service. Take steps to ensure that PrizmDoc Server is not accessible to end-users or the public internet by configuring a firewall in front of PrizmDoc Server to block access to the port it is using.

Ports

The following default ports should be open to access PrizmDoc Server:

Single-server Mode

- **18681** – PrizmDoc Server Entry Point (SEP) default port

Cluster Mode

- **18681** – PrizmDoc Server Cloud Entry Point (CEP) default port
- **18682** – PrizmDoc Server Entry Point (SEP) default port

NOTE: *PrizmDoc Server uses a number of ports for internal purposes and must not be accessible from outside of the server.*

PrizmDoc Server Administration

PrizmDoc Server includes a [Health Status API](#) to request real-time information about the state and health of the system. A sample [ASP.NET web application](#) is also included in the Windows installation that takes advantage of the Health Status API and demonstrates potential use cases.

The Health Status API provides information that can be helpful in diagnosing problems. However, it also contains sensitive information such as document information and specific processing tasks. Because of this, the ASP.NET WebForms sample or any application accessing the Health Status API of PrizmDoc Server should not be accessible to end-users or the public internet.

Secure Viewing Sessions

The [central configuration](#) file contains properties that can help prevent users from setting inappropriate values to try and attack the PrizmDoc Server, which could render performance problems with the server. These values are properties in the **ViewingSessionProperties** object that a client-user passes to PrizmDoc Server to start a viewing session.

The file paths for the Central Configuration file are:

- **Linux:** /usr/share/prizm/prizm-services-config.yml
- **Windows:** C:\Prizm\prizm-services-config.yml

NOTE: *The default installation directory is: C:\Prizm.*

For more information on creating secure viewing sessions, refer to the following topics:

- [Enabling Content Encryption](#)
- [Viewing Sessions API](#)

The following configuration properties put limits on viewing session properties sensitive to abusive attacks:

Central Configuration Properties Example

```
# Defines the min and max allowed values for the countOfInitialPages viewing
# session creation option.
#
# viewing.sessionConstraints.countOfInitialPages.min: 0
# viewing.sessionConstraints.countOfInitialPages.max: 10

# A regex which defines the pattern of an acceptable value for the
# documentExtension viewing session creation option.
#
# viewing.sessionConstraints.documentExtension.regex: ".\*"

# A regex which defines the pattern of an acceptable value for the
# externalId viewing session creation option.
#
# viewing.sessionConstraints.externalId.regex: ".\*"

# Defines the list of allowed values for the serverCaching viewing session
# creation option.
#
# Must be an array with one or more of the following strings:
#
# "none" - Allow REST API callers to create a new viewing session with caching
#         explicitly disabled.
#
# "full" - Allow REST API callers to create a new viewing session with caching
#         explicitly enabled.
#
# viewing.sessionConstraints.serverCaching.allowedValues: \["none","full"\]

# Defines the list of allowed values for the alwaysUseRaster viewing session
# creation option.
#
# Must be an array with one or more of the following values:
#
# false - Allow REST API callers to create a new viewing session which will
#         generate both raster and vector page content. Ideal for modern
#         browsers.
#
# true - Allow REST API callers to create a new viewing session which will
#        only generate raster content; vector content will not be generated.
#        This is useful for some older browsers.
#
# viewing.sessionConstraints.render.alwaysUseRaster.allowedValues: \[false\]
```

Server Side Request Forgery Concerns

Some source files, such as HTML, email, and Microsoft Office files, reference content that resides on another server. When rendering or converting these kinds of files, PrizmDoc Server may make network requests for this external content. Depending on what your deployment servers have network access to, this can be a security vulnerability. An attacker may leverage this to gain access to internal data and/or cause undesirable behavior. We recommend you take steps to prevent this.

There are two primary kinds of files you should be concerned with: 1) HTML and email files and 2) Microsoft Office files.

HTML and Email Files

HTML code can be found in both HTML and email (EML and MSG) files. When rendering or converting these files, the content may include an HTML tag (like an image or iframe) which refers to external content. The presence of something like an iframe tag could allow an attacker a way to gain access to *any* URL or HTTP resource in your network, potentially exposing a variety of data you never intended to.

If you do not need to render or convert HTML or email files, or if you do but you never need to include external content, we recommend you block all requests for external content by setting [security.htmlRendering.blockExternalContent](#) to **true**. When you do this, PrizmDoc Server will avoid making any external requests when rendering or converting HTML and email files.

Microsoft Office Files

When rendering or converting Office files, both PrizmDoc Server's built-in LibreOffice and Microsoft Office on Windows may make external requests for image data.

Additionally, Microsoft Excel may make external requests for text data. If you are using Microsoft Office for rendering, we recommend you disable the Excel WEBSERVICE function:

- Microsoft Office 2013: change the [policy](#) value for User Configuration -> Administrative Templates -> Microsoft Excel 2013 -> Excel Options -> Security -> "WEBSERVICE Function Notification Settings" to "Disabled".
- Microsoft Office 2016 / 2019: change the [policy](#) value for User Configuration -> Administrative Templates -> Microsoft Excel 2016 -> Excel Options -> Security -> "WEBSERVICE Function Notification Settings" to "Disabled".

Firewall Rules

In addition to the recommendations above, we also recommend you put in place firewall rules to prevent unintended access to hosts in your network.

Windows

On Windows, we recommend setting up per-process firewall rules.

For HTML and email files, the following PrizmDoc Server executables may make requests for external content:

- C:\Prizm\modules\wkhtmltopdf.exe
- C:\Prizm\modules\wkhtmltoimage.exe

When using Microsoft Office, the following executables are known to potentially request external content:

- C:\Program Files\Microsoft Office\Office16\EXCEL.EXE
- C:\Program Files\Microsoft Office\Office16\WINWORD.EXE
- C:\Program Files\Microsoft Office\Office16\POWERPNT.EXE
- C:\Program Files\Microsoft Office\Office15\WINWORD.EXE
- C:\Program Files\Microsoft Office\Office15\EXCEL.EXE
- C:\Program Files\Microsoft Office\Office15\POWERPNT.EXE

When using PrizmDoc Server's built-in LibreOffice, the following executable may make requests for external content:

- C:\Prizm\libreoffice\program\soffice.bin

If you do not need to render external content, we recommend you set up firewall rules which prevent all of the above executables from being able to make outgoing network requests.

However, if you need to allow rendering of some external content, we still recommend you set up firewall rules which only allow these executables to access hosts you consider to be safe or, at the very least, prevent access to

all internal hosts an attacker should not have access to (like metadata services, typically running on 169.254.169.254, or any other sensitive internal services or servers).

Linux

On linux, you cannot set up per-process firewall rules, but there are a variety of ways you can prevent PrizmDoc Server from making unintended network requests (user-specific firewall rules, Docker networking rules, and more). How exactly you set this up will depend upon your environment, but we recommend putting some sort of firewall rules in place which prevent PrizmDoc Server from being able to make network requests to hosts an attacker should not have access to.

Document Rendering Specifics

MSO Rendering Engine

When rendering MS Word documents containing comments with MSO add-on, the following specifics should be taken into account:

- If there are a lot of comments on the page that cannot be fully fit into the sidebar on a single page, some of the comments might not be rendered.
- If there are comments containing a long text, they are rendered partially and appended by a 3 dot ellipse, indicating the truncated state of the comment.

This is expected MS Office behavior when exporting document content with a large number of comments located on the same page, or comments containing a long text.

Meeting Information in Email Files

When the experimental [central configuration](#) option for rendering meeting information associated with email files is enabled, the following additional headers can be rendered by PrizmDoc Viewer:

- `When` - describes the meeting date-time and recurrence rule. PrizmDoc Viewer supports the basic recurrence settings and does not support all of the custom complex variations. Here is a list with examples of which occurrence settings are supported:
 - `Daily`. Example: `Occurs every day from March 13, 2021 2:00 PM - 2:15 PM (UTC+07:00)`
 - `Weekly on specified weekday(s)`. Example: `Occurs every Sunday, Monday, Saturday from March 13, 2021 5:00 PM - 6:00 PM (UTC+07:00)`
 - `Monthly on specified day`. Example: `Occurs day 13 of every month from March 13, 2021 2:00 PM - 3:00 PM (UTC+07:00)`
 - `Monthly on n-th weekday`. Example: `Occurs the third Saturday of every month from March 20, 2021 7:00 PM - 8:00 PM (UTC+07:00)`
 - `Annually on specified month and day`. Example: `Occurs every year from March 13, 2021 4:00 PM - 5:00 PM (UTC+07:00)`

The recurrence interval, end date, occurrence count, and other more complex rules are not supported. When the recurrence rule is not supported, the following message is displayed `"this event has a recurrence rule that cannot be displayed"`.

- `Who` - lists meeting attendees.

NOTE: This feature is a work-in-progress that is not officially supported by Accusoft. Its behavior may change at any time

in a future release of the product. We are collecting and reviewing any feedback you can provide about this feature at <https://ideas.accusoft.com/ideas/PDV-I-745>.

Installing

This section covers how to deploy an instance of PrizmDoc Server:

- [Using Docker](#)
- [Installing on Windows](#)
 - [Windows Requirements & Supported Environments](#)
 - [Install on Windows](#)
 - [Registry Changes](#)
 - [Natively Render MSO Documents](#)
 - [Unattended Install & Uninstall](#)
 - [Uninstall PrizmDoc Server on Windows](#)
- [Installing with Traditional Linux Install Packages \(deprecated\)](#)
 - [Requirements & Supported Environments for Traditional Linux Install Packages](#)
 - [Install Using Traditional Linux Install Packages](#)
 - [Install Asian Fonts on Traditional Linux Install Packages](#)
 - [Install Using Traditional Linux Install Packages on a Headless Environment](#)
 - [Uninstall Traditional Linux Install Packages](#)
- [Checking PrizmDoc Server Health](#)
- [Upgrade PrizmDoc Viewer](#)
- [Configuring a Cluster](#)

Using Docker

Using Docker

This section explains how to deploy a PrizmDoc Server instance using the official PrizmDoc Server docker image, available on Docker Hub as [accusoft/prizmdoc-server](#).

Requirements

To run PrizmDoc Server as a Docker container, you simply need a Docker host (a machine with Docker installed). See the [Docker documentation](#) for more information.

1. Create a PrizmDoc Server config file

Before you can run PrizmDoc Server, you'll need a config file. We've included a special `init-config` command in our Docker image which you can use to create an initial config file.

Windows (PowerShell)

First, make sure you've created a `config` directory on your host file system. This will be the directory where your new config file will be created:

```
mkdir config
```

Then, use the Docker image's `init-config` command to create a new PrizmDoc Server config file:

```
docker run --rm -e ACCEPT_EULA=YES --volume $pwd/config:/config
accusoft/prizmdoc-server init-config
```

This will create a new PrizmDoc Server config file on your Windows host filesystem at `.\config\prizm-services-config.yml`.

Linux (bash)

Use the `init-config` command to create a new PrizmDoc Server config file:

```
docker run --rm -e ACCEPT_EULA=YES --volume $(pwd)/config:/config
accusoft/prizmdoc-server init-config
```

This will create a new PrizmDoc Server config file on your host filesystem at `./config/prizm-services-config.yml`.

2. Configure your license

To start an instance of PrizmDoc Server, you need a license key from Accusoft. If you don't have a license, please contact info@accusoft.com.

To configure the license, set the values of `license.solutionName` and `license.key` in your PrizmDoc Server config file (`prizm-services-config.yml`).

NOTE: On a Linux system, because the config file was created by a Docker container, you will need to either edit the config file as root or change the owner of the file before editing it.

For more information about configuring PrizmDoc Server, see [Configuring PrizmDoc Server](#).

3. Start PrizmDoc Server

If you are using the default configuration, you can start PrizmDoc Server like so:

Windows (PowerShell)

First, create a directory on your host file system to store log files:

```
mkdir logs
```

Then, start a `prizmdoc-server` container:

```
docker run --rm --env ACCEPT_EULA=YES --publish 18681:18681 --volume
$pwd/config:/config --volume $pwd/logs:/logs --name prizmdoc-server
accusoft/prizmdoc-server
```

Linux (bash)

Start a `prizmdoc-server` container:

```
docker run --rm --env ACCEPT_EULA=YES --publish 18681:18681 --volume
$(pwd)/config:/config --volume $(pwd)/logs:/logs --volume $(pwd)/data:/data --
name prizmdoc-server accusoft/prizmdoc-server
```

In the examples above:

- `--rm` ensures the container is automatically deleted when it stops.
- `--env ACCEPT_EULA=YES` indicates you have accepted the [PrizmDoc Viewer license agreement](#).
- `--publish 18681:18681` publishes the container's `network.publicPort` port to the host. If you enable clustering (`network.clustering.enabled: true`), you will also want to publish the `network.clustering.clusterPort` (18682 in a default configuration).
- `--volume $(pwd)/config:/config` maps a host config directory into the container. Your local config directory must contain the `prizm-services-config.yml` config file created earlier.
- `--volume $(pwd)/logs:/logs` maps a local logs directory into the container. After the container stops, the logs will remain in this directory.
- `--volume $(pwd)/data:/data` (Linux only) maps a local data directory into the container. After the container stops, the data will remain in this directory. Because PrizmDoc Server uses memory-mapped files when writing to the data directory, and because Docker only supports memory-mapped file access to a mapped volume on a Linux host, **setting up this particular volume is only supported on a Linux docker host**.
- `--name prizmdoc-server` sets the name of the running container.
- `accusoft/prizmdoc-server` is the image that should be run.

If you want to start the Docker container in the background, add the `-d` option to `docker run` to run the container in disconnected mode.

4. Check PrizmDoc Server has finished starting and is healthy

It may take several minutes for PrizmDoc Server to finish starting. But, once fully started, `GET /PCCIS/V1/Service/Current/Health` should return HTTP 200, indicating PrizmDoc Server is healthy (while starting, this request will return nothing or an error).

Windows (PowerShell)

```
Invoke-WebRequest -Uri http://localhost:18681/PCCIS/V1/Service/Current/Health
```

Should eventually output something like:

```
StatusCode      : 200
StatusDescription : OK
Content         : {79, 75}
RawContent      : HTTP/1.1 200 OK
                  X-Server-Chain: 2b95007f011f
                  X-Server-Response-Time: 0
                  Connection: keep-alive
                  Transfer-Encoding: chunked
                  Date: Thu, 21 Nov 2019 17:01:04 GMT
Headers         : {[X-Server-Chain, 2b95007f011f], [X-Server-Response-Time, 0],
                  [Connection, keep-alive], [Transfer-Encoding, chunked]...}
```

```
RawContentLength : 2
```

NOTE: It may take several minutes for PrizmDoc Server to start. You will not get a 200 result until PrizmDoc Server has fully started.

Linux (bash)

```
curl -i http://localhost:18681/PCCIS/V1/Service/Current/Health
```

Should eventually output something like:

```
HTTP/1.1 200 OK
X-Server-Chain: 69a6d22f90b9
X-Server-Response-Time: 1
Date: Wed, 20 Nov 2019 20:25:31 GMT
Connection: keep-alive
Transfer-Encoding: chunked

OK
```

NOTE: It may take several minutes for PrizmDoc Server to start. You will not get a 200 result until PrizmDoc Server has fully started.

Troubleshooting: Check the admin status page

If you're not getting a 200 response at all, try checking the PrizmDoc Server admin status page in a browser:
<http://localhost:18681/admin>

5. Stopping the container

You can stop your named container with:

```
docker stop prizmdoc-server
```

Install on Windows

This section contains the following information:

- [Windows Requirements & Supported Environments](#)
- [Install on Windows](#)
- [Registry Changes](#)
- [Natively Render MSO Documents](#)
- [Unattended Install & Uninstall](#)
- [Uninstall PrizmDoc Server on Windows](#)

Windows Requirements & Supported Environments

Introduction

This section provides information about the system requirements for PrizmDoc Server when using the PrizmDoc Viewer Self-Hosted installation:

TIP: PrizmDoc Viewer is only supported on 64-bit operating systems.

PrizmDoc Server

PrizmDoc Server is a suite of RESTful APIs that control document conversion processes. It requires significant memory and processing power.

Supported Operating Systems

Windows

NOTE: When using Windows Server 2016 or Windows Server 2019, you must use the Desktop Experience version. The core version doesn't contain the components needed for PrizmDoc.

- Windows Server 2012, 2012 R2
- Windows Server 2016, 2019 with Desktop Experience

System Requirements

Windows Microsoft .NET Framework 4.0

Hardware Requirements

Requirements vary greatly based on usage and it is generally a good idea to find what best fits your expected usage. The [Sizing Guide](#) is a good place to start understanding what resources PrizmDoc Server uses and how to optimize them for your needs.

Requirements

.NET 4.0 Framework

In order to install the PrizmDoc Server on a Windows system, the following item needs to be installed and available before the installer is run:

- **.NET 4.0 Framework:** The Windows Service that starts PrizmDoc Viewer and the PrizmDoc Server are built targeting the .NET 4.0 framework.

Microsoft Office Conversion

In order to use the Microsoft Office Conversion (MSO) rendering feature on a Windows system, the following components are required to be installed and available before the installer is run:

- Microsoft Office 2013, 2016, or 2019 (not included in the PrizmDoc Server distribution and licensed separately) and the corresponding Windows Updates (Please refer to <https://www.microsoft.com/en-us/microsoft-365/microsoft-365-and-office-resources> in order to get system requirements for MS Office versions).

- [Microsoft XPS Document Writer](#) printer driver.
- [Ink and Handwriting Services](#) feature from the Server Manager.

NOTE: The installed copy of Microsoft Office **must be activated** in order for PrizmDoc Viewer's Microsoft Office Conversion Service to work properly. If the installed copy of Microsoft Office is not licensed, not activated, or an expired or trial version, then Microsoft Office will not work with PrizmDoc Viewer. Also note that subscription-based Microsoft Office versions like Office 365 must be downloaded as a desktop version in order to work with PrizmDoc Viewer.

NOTE: If you are running PrizmDoc Viewer in cluster mode, **all servers** in the Windows MSO cluster **must use the same version** of Microsoft Office.

NOTE: When considering using Microsoft Office 2019 as a rendering engine for Excel source documents, please be aware that its performance could be up to three times slower compared to the Excel performance of Microsoft Office 2016. So, we recommend using Microsoft Office 2019 when its new [features](#) are required or for Windows version compatibility, otherwise prefer Microsoft Office 2016 for better Excel rendering performance if possible.

Install on Windows

Windows Installation

This section covers everything you need to install PrizmDoc Server. If you have questions about requirements before installing, refer to the [System Requirements & Supported Environments](#) topic.

Registry Changes

The installer may make changes to the registry and require a reboot, if the registry is not pre-configured. For more information about this, see [Registry Changes](#) for more detailed information.

Cloud Deployment

Before saving an image of a system, onto which a future non-interactive installation of PrizmDoc Server will occur, perform the [Before Installation](#) steps (below) and [configure the registry](#) manually.

Before Installation

These actions must be taken before PrizmDoc Server is installed, regardless of whether the installation is interactive or non-interactive.

IMPORTANT: Make sure you **back up your configuration files before upgrading** from any previous versions of PrizmDoc.

NOTE: If you have an updated license, you must re-start PAS and PrizmDoc Server in order to use the new license.

All the required components listed below must be installed and manually configured prior to installation.

Configure Rendering With Microsoft Office

If your license includes the MSO Conversion feature, which allows the server to render Microsoft Office documents natively with Microsoft Office, follow the pre-installation steps to [Natively Render Microsoft Office Documents](#).

Installation

To install PrizmDoc Server on your own Windows server, follow these steps:

1. Download **PrizmDoc Server** from the [website](#).
2. Double-click on the **PrizmDocServer-xx** application file to launch the installer (where xx represents the version). Click **Install PrizmDoc Server**.
3. Carefully read the information contained in the License Agreement form before making a decision to accept the terms of the License Agreement. Choose **I accept the terms in the License Agreement** to accept the conditions outlined in the License Agreement and then click the **Next** button to continue the installation (or click **Cancel** to exit the installation process).
4. If your machine does not meet the [PrizmDoc Server minimum system requirements](#), a dialog displays indicating the requirements that are not met. (You may choose to continue installation, but may experience poor performance.) Click **Next**.
5. The Installation Path dialog is displayed. Specify the **destination directory** where the PrizmDoc product should be installed or choose the default installation destination directory, then click **Next**.
6. Specify the **login account** (account name and password) that PrizmDoc Server will run under. Please note that the "login account" should have enough privileges to start Windows Service - it should have **Log on as a service** user right. If you are using the Microsoft Office Conversion (MSO) add-on, please make sure that the "login account" is a real user account with **Administrator** rights. Running PrizmDoc under the LocalSystem user or another Microsoft Windows integrated service account is not supported for this option. Please consider the cases below for the "login account" and the Microsoft Office installation account:
 - The "login account" is **the same** administrative account that has been used for the installation of Microsoft Office on the system - no additional steps are required.
 - The "login account" is **not the same** administrative account that has been used for the installation of Microsoft Office on the system. Before installing PrizmDoc, make sure to log into the "login account" and run Microsoft Word, Excel, and PowerPoint applications once so that the corresponding registry keys for Microsoft Office are created for the "login account".
7. Click **Install** to continue. The Installation dialog is displayed with a progress bar. While PrizmDoc Server is installing, you can click on the links to review the **Release Notes**, **Online Demos** or **Code Examples**.
8. Part way through the installation, the installer will launch the PrizmDoc Licensing Utility, a tool which allows you to either install a paid license or configure the product for evaluation.
 - If you are evaluating the product, simply select **I want to evaluate the product**.
 - If you have a paid license, select **I have purchased a license** and follow the instructions on the following screens to install your license.
9. Once you have finished installing your license or configuring the product for evaluation, click the **Exit** button to close the PrizmDoc Licensing Utility and return to the installation process.
10. Once complete, the Installation Complete dialog is displayed. If a reboot is necessary to finalize the installation, a **Restart Now** checkbox will be present. By default this option will be checked, indicating the installer will automatically restart your machine when you click **finish** (if you would prefer to delay this restart and perform it yourself, you can uncheck the **Restart Now** option before clicking **finish**, but you **MUST** restart the machine before using PrizmDoc Server). Click **finish** to exit the installer.
11. If you purchased PrizmDoc with the Microsoft Office Conversion service, continue to the next section below. If you did not purchase PrizmDoc with the Microsoft Office Conversion service, then your installation is

complete.

12. It is important to block specific Windows firewall ports to prevent SMB traffic from leaving the configured Windows instance. The following outbound ports should be blocked: TCP: 137, 138, 139, 445 and UDP: 137, 138. Please refer to the [Microsoft guidelines](#) for blocking specific firewall ports to prevent SMB traffic.
13. It is important to block all outbound traffic of Microsoft Excel application in Windows firewall. Please refer to the [Microsoft guidelines](#) for blocking of application outbound traffic.

NOTE: For a production installation of PrizmDoc on Windows, you can improve your performance by enabling Gzip compression.

Registry Changes

Registry Changes

The PrizmDoc installer will modify the registry to increase the size of the non-interactive heap, allowing the Prizm service to create more than the default number of child processes to run its micro-services.

If the registry already contains a value greater than necessary, no changes to the registry will be made. If the registry is modified, and PrizmDoc is un-installed, the original registry values will be replaced.

NOTE: Upon start up, the PrizmDoc Server checks whether or not the system's non-interactive heap size corresponds to the CPU cores count, and in the case of a discrepancy, the Admin page will never show a 'Running' status, and the OfficeConversionService.log will contain the error message: "Non-interactive heap size does not correspond to CPU cores count, going to 'Unhealthy' state". Please follow the recommendations provided in the next two sections below to avoid server configuration errors related to a mismatch between the non-interactive heap size and the CPU cores count.

Interactive Installation

A manual heap size value adjustment process is not required when the PrizmDoc Windows Server installer is running in its interactive UI mode, where it will make all of the necessary registry changes and require a system reboot.

Please make sure to install the product on the system, with the **final** CPU cores count to be used in production, so that the product installer will adjust the non-interactive heap size accordingly, corresponding to the actual CPU cores count.

Due to a known issue with the installer, it might not ask for a reboot after the installation, so please make sure to reboot the system manually. This issue will be addressed in the future release.

Non-Interactive Installation

In order to avoid a system reboot when installing PrizmDoc Server for Windows in non-interactive mode, such as for cloud-based systems using auto-deployment, the cloud image must have its registry modified as described below.

Registry Changes

Registry Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\SubSystems

Value: Windows

The value is a complex, space separated set of system wide settings, one of which is named "**SharedSection**". This is a comma-separated string value containing 3 numeric values that may look like: SharedSection=1024,20480,768.

The third numeric value is a non-interactive desktop heap size in kilobytes, which must be set to the Heap Size Value (see table below) corresponding to the number of CPU cores in the server:

CPU Cores	Instance Count	Heap Size Multiplication Factor	Heap Size Value
1	1	1	768 (default)
2, 3	Number of Cores	2	1536
4 <= Cores < 8	4	2	1536
8	8	3	2304
12	12	4	3072
16	16	5	3840
20	20	6	4608
24	24	7	5376
28	28	8	6144
32	32	9	6912

The heap size value is calculated from the default size by applying a heap size multiplier corresponding to the number of CPU cores available on the server.

The instance count indicates the number of **concurrent** Office document converter processes the Office Conversion Service will handle when using the MSO option. The total number of running Office document converter processes will be twice the number of concurrent instances - this is done in order to improve the performance of the Office Conversion Service.

NOTE: The PrizmDoc Server limits the max CPU core usage to 32 for Office documents processing. If your machine has more CPU cores, the extra ones will not be utilized to avoid server's overload and instability - this is related to the system resources usage by MSO. So if you need to utilize more CPU power for Office documents viewing and conversion, please consider using [PrizmDoc Server Clustering](#).

Natively Render MSO Documents

Natively Render Microsoft Office Documents

Accusoft offers a Microsoft Office Conversion (MSO) add-on option for PrizmDoc Server running on Windows. The Microsoft Office Conversion service allows you to have native rendering of Microsoft documents in PrizmDoc's Viewer, Content Conversion and MarkupBurner services. Available as an add-on licensed option to your PrizmDoc Windows product, this enhanced rendering offers a solution for companies who have a business requirement of native rendering of Microsoft Office documents. Please refer to [Supported File Formats](#) section for complete list of document formats supported by MSO.

The MSO conversion option is triggered by a license key that includes the MSO feature. The MSO feature can be purchased in addition to the standard features of your PrizmDoc Server license.

Prerequisites Before Installing PrizmDoc Server

Before installing PrizmDoc Server, make sure you've done the following:

1. **Ensure the Microsoft XPS Document Writer printer is installed.** For Microsoft Office rendering, PrizmDoc Server requires the **Microsoft XPS Document Writer** printer driver to be installed (it does not need to be the default printer, but it does need to be installed and available).
2. **Ensure that Ink and Handwriting Services are enabled (for versions of Windows Server prior to v2016).** For Microsoft Office rendering, PrizmDoc Server requires the "Ink and Handwriting Services" desktop experience feature be enabled from the **Server Manager**.
3. **Ensure Microsoft Office is activated.** For Microsoft Office rendering, PrizmDoc Server requires your installed copy of Microsoft Office **licensed and activated**. If Microsoft is not licensed, not activated, or an expired or trial version, Microsoft Office rendering with PrizmDoc Server will not function.
4. **Ensure you run PrizmDoc Server with the same Windows user account you used to install Microsoft Office.** When installing PrizmDoc Server, you will be asked which Windows **login account** (account name and password) PrizmDoc Server should use when it is running. We recommend you use the same Windows user account which you used to install Microsoft Office.
 - o If you choose to run PrizmDoc Server with a *different* Windows user account than the user you used to install Microsoft Office, you will need to take an additional step before installing PrizmDoc Server: You must login using the Windows account you used to install Microsoft Office and run Microsoft Word, Excel, and PowerPoint each one time to ensure that Office applications will run smoothly when PrizmDoc Server uses them.
5. **Ensure that all PrizmDoc Server instances are using the same version of Microsoft Office.** If you are running PrizmDoc Server in clustered mode, **all servers** in the Windows MSO cluster **must use the same version** of Microsoft Office.
6. **Ensure that the Printer Spooler service is enabled.** The Windows Print Spooler service **must be enabled** for the **login account** in order for the Microsoft XPS Document Writer printer to work properly.

Unattended Install & Uninstall

Introduction

The PrizmDoc Server Windows installer can be installed unattended, however certain properties must be set:

IMPORTANT: *PrizmDoc requires a clean installation when migrating from a version earlier than v12.0. You must first uninstall any previous versions of PrizmDoc and reboot your system. Only then should you install PrizmDoc v12.0 or later. Make sure you back up your configuration files before uninstalling any previous versions of PrizmDoc. Once you have installed v12.0, you do not need to uninstall if you want to migrate to v12.1 or later.*

Property	Description	Default
ServiceUser	Required - The service account user name. This defines what user the PrizmDoc Server should run as. It should be in the format DOMAIN\USER. If you are using the Microsoft Office Conversion (MSO) add-on, please make sure that the required "ServiceUser" parameter value corresponds to a real user account with Administrator rights. Running PrizmDoc Server under the LocalSystem user or another Microsoft Windows integrated service account is not supported for this option.	None
ServicePassword	Required - The password for the ServiceUser.	None

Property	Description	Default
InstallFolder	Optional - The base installation directory for the product.	"C:\Prizm"

Unattended Install

To start the unattended install:

1. The PrizmDocServer.exe can be used to launch the installer and specify the above parameters in silent mode. Open a command line prompt as an Administrator, change to the folder where the .exe is located, and run the following (note that the values shown below for ServiceUser and ServicePassword are examples, and you will need to change them to specify your ServiceUser and ServicePassword):

Example

```
> PrizmDocServer.exe ServiceUser=accusoft.com\PrizmUser  
ServicePassword=pdpassword -s -l output.log
```

NOTE: The *-s* flag is required to trigger silent mode and prevent the UI from opening. Leaving this out will open the UI.

The *-l output.log* flag is optional. If specified, it will output a log of the entire install process to a file using the specified name for the filename. For a complete install, this will output 3 files. If the install fails, include these files in bug reports.

2. You may wish to run this with the start command to wait for completion, otherwise the install will start in the background and on a console or script, it will return immediately.

Example

```
> start /wait PrizmDocServer.exe ServiceUser=accusoft.com\PrizmUser  
ServicePassword=pdpassword -s -l output.log
```

3. If you have a paid license, configure your license (see [Licensing](#) for more information). If you are evaluating the product, you can skip this step.
4. Finally, start the service. See the [Getting Started with PrizmDoc Viewer > Starting & Stopping PrizmDoc Server > Windows](#) for more information on stopping and starting the service.

Example

```
> net start prizm
```

5. PrizmDoc Server should now be installed, licensed and started.

NOTE: When installing from the command line on Windows, the use of 8.3 notation to specify the install directory is not supported. While this may result in an error free install, some services may not start as expected.

Silent Uninstall

You can use the -u flag as shown below to silently uninstall PrizmDoc Server on Windows:

Example

```
> PrizmDocServer.exe -s -u -l output.log
```

Uninstall PrizmDoc Server on Windows

To uninstall Prizm from your Windows system, please perform the following steps:

1. Run **Add/Remove Programs**, or select **Programs and Features** from the Control Panel, or **Apps & features** from Settings.
2. Choose **PrizmDoc Server** from the list of installed applications.
3. Click the **Uninstall** button.
4. You will be prompted to confirm your desire to remove the product, answer affirmatively to continue.
5. The Uninstaller will start and remove PrizmDoc Server from your system.

If a reboot is necessary to completely uninstall PrizmDoc Server, a **Restart Now** checkbox will be present. By default this option will be checked, indicating the uninstaller will automatically restart your machine when you click **finish** (if you would prefer to delay this restart and perform it yourself, you can uncheck the **Restart Now** option before clicking **finish**, but you **MUST** restart the machine for PrizmDoc Server to be completely uninstalled). Click **finish** to exit the uninstaller.

Installing with Traditional Linux Install Packages (deprecated)

DEPRECATION NOTICE: While we currently continue to offer and support traditional Linux packages for direct installation on a Linux host, these have largely become obsolete now that Docker deployment is an option. **We have announced deprecation of our traditional Linux install packages** and, in a future product release, we intend to only offer our [Docker-based deployment option](#).

This section contains the following information:

- [Requirements & Supported Environments for Traditional Linux Install Packages](#)
- [Install Using Traditional Linux Install Packages](#)
- [Install Asian Fonts on Traditional Linux Install Packages](#)
- [Install Using Traditional Linux Install Packages on a Headless Environment](#)
- [Uninstall Traditional Linux Install Packages](#)

Requirements & Supported Environments for Traditional Linux Install Packages

DEPRECATION NOTICE: While we currently continue to offer and support traditional Linux packages for direct installation on a Linux host, these have largely become obsolete now that Docker deployment is an option. **We have announced deprecation of our traditional Linux install packages** and, in a

future product release, we intend to only offer our [Docker-based deployment option](#). The rest of this topic applies to traditional Linux install packages only.

Introduction

PrizmDoc Server is a suite of RESTful APIs that control document conversion processes. It requires significant memory and processing power.

Hardware Requirements

Requirements vary greatly based on usage and it is generally a good idea to find what best fits your expected usage. The [Sizing Guide](#) is a good place to start understanding what resources PrizmDoc Server uses and how to optimize them for your needs.

Supported Linux Distributions

64-bit editions of:

- CentOS 7
- Red Hat Enterprise Linux 7
- Ubuntu 18.04 LTS

Required Libraries

- libbz2.so.1
- libc.so.6
- libcairo.so.2
- libcups.so.2
- libdbus-glib-1.so.2
- libdl.so.2
- libexpat.so.1
- libfontconfig.so.1
- libfreetype.so.6
- libgcc_s.so.1
- libgif.so.4
- libGL.so.1
- libjpeg.so.62
- libm.so.6
- libnsl.so.1
- libopenjpeg.so.2
- libpixman-1.so.0
- libpng12.so.0
- libpthread.so.0
- librt.so.1
- libstdc++.so.6
- libthread_db.so.1
- libungif.so.4
- libuuid.so.1
- libX11.so.6
- libXau.so.6
- libxcb.so.1
- libXdmcp.so.6

- libXext.so.6
- libXi.so.6
- libXinerama.so.1
- libxml2.so.2
- libXrender.so.1
- libXtst.so.6
- libz.so.1
- linux-vdso.so.1

NOTE: PrizmDoc Server requires x86-64 versions of the libraries listed above.

To verify that the required libraries are installed, use "ldconfig" as shown in the following example:

Example

```
\# ldconfig -p | grep libcairo
libcairo.so.2 (libc6,x86-64) => /lib64/libcairo.so.2
```

Required Libraries (Ubuntu 18.04)

The following packages are included in the prizmdoc_server<version>.amd64.deb.tar.gz package which is located in the 'packages' subfolder:

- libgif4_4.1.6-11+deb8u1_amd64.deb
- libopenjpeg2_1.3+dfsg-4.8_amd64.deb

Install Using Traditional Linux Install Packages

Introduction

DEPRECATION NOTICE: While we currently continue to offer and support traditional Linux packages for direct installation on a Linux host, these have largely become obsolete now that Docker deployment is an option. **We have announced deprecation of our traditional Linux install packages** and, in a future product release, we intend to only offer our [Docker-based deployment option](#). The rest of this topic applies to traditional Linux install packages only.

If you have questions about requirements before installing, refer to the [System Requirements & Supported Environments](#) topic.

IMPORTANT: PrizmDoc **requires a clean installation** when migrating from a version earlier than v12.0. You must first uninstall any previous versions of PrizmDoc and reboot your system. Only then should you install PrizmDoc v12.0 or later. **Make sure you back up your configuration files before uninstalling any previous versions of PrizmDoc.** Once you have installed v12.0, you do not need to uninstall if you want to migrate to v12.1 or later.

NOTE: PrizmDoc relies on a fontconfig package that is not shipped with the product and that might be missing from some distributions of Ubuntu. This was resolved by adding automated checks in the PrizmDoc 13.0 installation scripts. As a workaround for older versions of PrizmDoc, we recommend installing the fontconfig package manually by using `sudo apt-get install fontconfig`.

NOTE: If you have an updated license, you must re-start PAS and PrizmDoc Server in order to use the new license.

- [Verify the System's Locale](#)
- [Step 1 - Download PrizmDoc Server](#)
- [Step 2 - Unpack and Install the Downloaded Archive](#)
- [Step 3 \(Optional\) - Including PrizmDoc Services to the Boot Sequence](#)
- [Step 4 - Configure](#)

- [Step 5 - Verify that the Installation was Successful](#)
- [How to Install Common Certificate Authority Root Certificates on Linux](#)

Some steps may be specific to a particular Linux distribution; these steps will be labeled as being specific to one of the following:

- "Red Hat / CentOS Linux Distributions"
- "Ubuntu Linux Distributions"

Make sure you log in as **root** to the machine.

Verify the System's Locale

1. To ensure your system's locale is specified, run the command:

Example

```
locale
```

2. If the LC_ALL entry is not set, you must specify it with the following:

Example

```
export LC_ALL="en_US.UTF-8"  
sudo localedef -v -c -i en_US -f UTF-8 en_US.UTF-8
```

Step 1 - Download PrizmDoc Server

IMPORTANT: Before you download PrizmDoc, note that packages are only available for 64-bit systems.

1. Download **PrizmDoc Server** from the [website](#) by selecting the desired Linux Distribution.
OR
2. Download directly to the Linux server using the 'wget' command for the specific distribution as shown below:

NOTES:

1. **You must substitute the version of the package you are using in the code examples below.** For example, if you are using v13.8, then specify "13.8" instead of "<version>". If the version is a hot fix, you will also need to specify the hot fix number, for example, "13.8.1".
2. **Instructions assume that 'wget' has already been installed** on the server OS.

Red Hat Enterprise Linux and CentOS v7 (and later)

Example

```
wget  
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_server_<version>.RHEL7.tar.gz
```

Ubuntu Linux Distributions

Example

```
wget  
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_server_<version>.amd64.deb.tar.gz
```

Generic .tar.gz Distribution

Example

```
wget
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_server_<version>.x86_64.tar.gz
```

For license questions, please contact info@accusoft.com.

Step 2 - Unpack and Install the Downloaded Archive

Open a command line and change to the location where you downloaded the tarball. Use the following command line examples appropriate for your distribution to:

1. Decompress and unpack the downloaded file. After you have unpacked the archive, the contents will have been decompressed into a directory named **prizmdoc_server_<version>.<arch>[.rpm|.deb]**.
2. Change to the unpacked directory and install the packages.

Red Hat, CentOS, and Older Linux Distributions

The following example is for Red Hat, CentOS, and older Linux distributions:

Example

```
tar -xzvf prizmdoc_server_*.tar.gz
cd prizmdoc_server_*
yum install --nogpgcheck *.rpm
```

Ubuntu Linux Distributions

The following example is for Ubuntu Linux distributions:

Example

```
tar -xzvf prizmdoc_server_*.deb.tar.gz
cd prizmdoc_server_*.deb

# Note for Ubuntu 18.04, you must run the following commands before installing
PrizmDoc.

sudo dpkg -i packages/*.deb
sudo dpkg -i *.deb

# Note that 'dpkg' does not resolve dependencies automatically, so please ignore
possible errors.
# If there are errors about missing dependencies, invoke the following commands to
install
# the dependencies and complete the configuration of the packages.

sudo apt-get update
sudo apt-get -f install
```

Generic .tar.gz Distribution

We also provide a generic .tar.gz package. Please review the [System Requirements and Supported Environments](#) topic to ensure compatibility. You will also need to install the **dependencies** described in the [Requirements](#) section. Once the dependencies are installed, you can install the **.tar.gz** with the following commands as root:

Example

```
tar -xzvf prizmdoc_server_*.tar.gz
cd prizmdoc_server_*
ls prizm-*.tar.gz | xargs -n1 tar xzf
cp -R prizm /usr/share/
```

Add symbolic links to the fonts directory and update the system fonts cache to enable the usage of installed fonts by PrizmDoc services.

Example

```
ln -s /usr/share/prizm/modules/poppler/fonts/accusoft_prizm_fonts.conf
/etc/fonts/conf.d/99-accusoft_prizm_fonts.conf
fc-cache -f
```

Step 3 (Optional) - Including PrizmDoc Services to the Boot Sequence

You can configure PrizmDoc Services to start/stop together with the system in two steps:

1. Create a symbolic link to **/usr/share/prizm/scripts/pccis.sh** in the **/etc/init.d/** directory:

Example

```
ln -s /usr/share/prizm/scripts/pccis.sh /etc/init.d/pccis
```

2. Register PrizmDoc Services as an init script, so that it is managed by the system. This step is platform-dependent.

Red Hat, Fedora, CentOS, and Older Linux Distributions

Example

```
chkconfig --add pccis
```

Ubuntu Linux Distributions

Example

```
update-rc.d pccis defaults
```

Once this is done, the system should stop PrizmDoc Services when rebooting or shutting down, and will start again when the server boots up.

Excluding PrizmDoc Services from the Boot Sequence

If you want to prevent PrizmDoc Services from starting/stopping together with the system, you need to revert Step 2 from

the section above. This can be performed as follows:

Red Hat, Fedora, CentOS, and Older Linux Distributions

Example

```
chkconfig --del pccis
```

Ubuntu Linux Distributions

Example

```
update-rc.d -f pccis remove
```

Step 4 - Configure

1. If you have a paid license, configure your license (if you are evaluating the product, you can skip this step). See [Licensing](#) for more information.
2. For a production installation, you will want to configure where log files are stored. See the Logging section in the [Central Configuration](#) topic.
3. If you are licensed to use the Microsoft Office Conversion add-on for PrizmDoc Servers running on Linux, you need to configure your server as described in the topic: [Configure Microsoft Office Conversion Connectivity](#).
4. Open ports in your firewall according to the [Security Guidance](#).

Step 5 - Verify that the Installation was Successful

1. If PrizmDoc Server is already running, **stop PrizmDoc Server:**

```
/usr/share/prizm/scripts/pccis.sh stop
```

2. **Start PrizmDoc Server:**

```
/usr/share/prizm/scripts/pccis.sh start
```

If you want PrizmDoc Server to automatically start on boot, see [Adding PCCIS to the Boot Sequence](#).

3. **Wait for PrizmDoc Server to become healthy.** Poll `http://yourserver:18681/PCCIS/V1/Service/Current/Health` with a `GET` request until it returns `HTTP 200`, indicating the server is healthy. See the [Health Status API](#) for more information.

Once PrizmDoc Server reports healthy, your installation is complete.

How to Install Common Certificate Authority Root Certificates on Linux

The following commands should all be run as root. Additionally, if prompted for addition/removal permission, then yes/no should be entered as the response.

There are a few options for installing SSL certificates. In all cases, the certificates are stored in `/usr/share/.mono/certs/`.

Install Mozilla's root CA certificates

You can do it with `cert-sync` tool, which synchronizes the Mono SSL certificate store against your OS certificate store:

Ubuntu Linux Distributions


```
/usr/share/prizm/mono/64/bin/cert-sync /etc/ssl/certs/ca-certificates.crt
```

Red Hat and CentOS Linux Distributions

```
/usr/share/prizm/mono/64/bin/cert-sync /etc/pki/tls/certs/ca-bundle.crt
```

NOTE: Your distribution might use a different path to the CA certificates file, if it's not derived from one of those.

Install the CA certificates that are provided by your operating system

Distributions will generally contain CA certificates already, to be used by various tools. They will be stores in a `ca-bundle.crt` file. You can import those certificates using mono's `certmgr` utility, as such:

```
awk 'BEGIN {c=0;} /BEGIN CERT/{c++} { print > "cert." c ".crt"}' < /etc/ssl/certs/ca-bundle.crt
find . -name 'cert.*.crt' -exec sudo /usr/share/prizm/mono/64/bin/certmgr -add -c -m Trust {} \;
```

NOTE: this will split the bundle file into individual certificate files and import each certificate. It is best to execute this from an empty directory in order to avoid file conflicts. The individual certificates can be deleted after the import.

Install individual certificate files

If you already have a list of custom certificates you need to trust, you can import each certificate file, as such:

```
/usr/share/prizm/mono/64/bin/certmgr -add -c -m Trust /path/to/certificatefile
```

Install Asian Fonts on Traditional Linux Install Packages

Introduction

DEPRECATION NOTICE: While we currently continue to offer and support traditional Linux packages for direct installation on a Linux host, these have largely become obsolete now that Docker deployment is an option. **We have announced deprecation of our traditional Linux install packages** and, in a future product release, we intend to only offer our [Docker-based deployment option](#). The rest of this topic applies to traditional Linux install packages only.

This section contains important information for installing Asian fonts:

- [Red Hat & CentOS](#)
- [Ubuntu](#)

Red Hat and CentOS

By default, Asian language support is not installed on the RHEL / CentOS systems. In order to properly render documents with Asian fonts, support for corresponding languages should be installed. There is a single Fonts package which includes support for these languages:

Example

```
# yum groupinstall Fonts
```

Ubuntu

By default, Asian language support is not installed on Ubuntu systems. In order to properly render documents with Asian fonts, support for corresponding languages should be installed.

To install Japanese language support, run following commands:

Example

```
# sudo apt-get install language-pack-ja
# sudo apt-get install japan*
```

To install Chinese language support, run following commands:

Example

```
# sudo apt-get install language-pack-zh
# sudo apt-get install chinese*
```

To install Korean language support, run following commands:

Example

```
# sudo apt-get install language-pack-ko
# sudo apt-get install korean*
```

And finally, you will need to add additional fonts:

Example

```
# sudo apt-get install fonts-arphic-ukai fonts-arphic-uming fonts-ipafont-mincho
fonts-ipafont-gothic fonts-unfonts-core
```

NOTE: If PrizmDoc Viewer was running when you installed language/font support, you must restart PrizmDoc Viewer in order to apply the changes.

Install Using Traditional Linux Install Packages on a Headless Environment

Install on a Headless Environment

DEPRECATION NOTICE: While we currently continue to offer and support traditional Linux packages for direct installation on a Linux host, these have largely become obsolete now that Docker deployment is an option. **We have announced deprecation of our traditional Linux install packages** and, in a future product release, we intend to only offer our [Docker-based deployment option](#). The rest of this topic applies to traditional Linux install packages only.

Use the following steps to install PrizmDoc in a Linux headless environment:

IMPORTANT: PrizmDoc **requires a clean installation** when migrating from a version earlier than v12.0. You must first **uninstall any previous versions of PrizmDoc and reboot your system**. Only then should you install PrizmDoc v12.0 or later. **Make sure you back up your configuration files before uninstalling any previous versions of PrizmDoc.** Once you have installed v12.0, you do not need to uninstall if you want to migrate to v12.1 or later.

NOTE: If you have an updated license, you must re-start PAS and PrizmDoc Server in order to use the new license.

- [Verify the System's Locale](#)
- [Step 1 - Download PrizmDoc](#)
- [Step 2 - Unpack and Install the Downloaded Archive](#)
- [Step 3 - Configure Your License](#)
- [Step 4 - Verify that the Installation was Successful](#)

Make sure you log in as **root** to the machine. All command lines preceded by the '>' sign are the example output of that command, where applicable.

Verify the System's Locale

1. To ensure your system's locale is specified, run the command:

Example

```
locale
```

2. If the LC_ALL entry is not set, you must specify it with the following:

Example

```
export LC_ALL="en_US.UTF-8"  
sudo localedef -v -c -i en_US -f UTF-8 en_US.UTF-8
```

Step 1 - Download PrizmDoc

IMPORTANT: Before you download PrizmDoc, note that packages are only available for 64-bit systems.

1. Download **PrizmDoc** from the [website](#) by selecting the desired Linux Distribution.
OR
2. Download directly to the Linux server using the 'wget' command for the specific distribution as shown below:

NOTES:

1. **You must substitute the version of the package you are using in the code examples below.** For example, if you are using v13.8, then specify "13.8" instead of "<version>". If the version is a hot fix, you will also need to specify the hot fix number, for example, "13.8.1".
2. **Instructions assume that 'wget' has already been installed** on the server OS.

Red Hat Enterprise Linux and CentOS v7 (and later)

Example

```
wget
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_server_<version>.RHEL7.tar.gz
```

Ubuntu Linux Distributions

Example

```
wget
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_server_<version>.amd64.deb.tar.gz
```

Generic .tar.gz Distribution

Example

```
wget
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_server_<version>.x86_64.tar.gz
```

For license questions, please contact info@accusoft.com.

Step 2 - Unpack and Install the Downloaded Archive

Open a command line and change to the location where you downloaded the tarball. Use the following command line examples appropriate for your distribution to:

1. Decompress and unpack the downloaded file. After you have unpacked the archive, the contents will have been decompressed into directories named: **prizmdoc_client_<version>.<arch>[.rpm|.deb]** and **prizmdoc_server_<version>.<arch>[.rpm|.deb]**.
2. Change to the unpacked directory and install the packages.

Red Hat, CentOS, and Older Linux Distributions

The following example is for Red Hat, CentOS, and older Linux distributions:

Viewer Example

```
tar -xzvf prizmdoc_client_*.tar.gz
cd prizmdoc_client_*
yum install --nogpgcheck *.rpm
```

Server Example

```
tar -xzvf prizmdoc_server_*.tar.gz
cd prizmdoc_server_*
yum install --nogpgcheck *.rpm
```

The Prizm installer does not install them automatically. Please **manually** download these packages and then install them using `--nogpgcheck` flag as follows:

Example

```
yum install --nogpgcheck ./openjpeg-libs-1.3-7.e15.x86_64.rpm
yum install --nogpgcheck ./pixman-0.22.0-2.2.e15_10.x86_64.rpm
```

Ubuntu Linux Distributions

The following example is for Ubuntu Linux distributions:

Viewer Example

```
tar -xzvf prizmdoc_client_*.deb.tar.gz
cd prizmdoc_client_*.deb
sudo dpkg -i *.deb
# 'dpkg' does not resolve dependencies automatically, so please ignore possible errors,
if you did not install required dependencies yet, and invoke next commands
sudo apt-get update
sudo apt-get -f install
```

Server Example

```
tar -xzvf prizmdoc_server_*.deb.tar.gz
cd prizmdoc_server_*.deb
sudo dpkg -i *.deb
# 'dpkg' does not resolve dependencies automatically, so please ignore possible errors,
if you did not install required dependencies yet, and invoke next commands
sudo apt-get update
sudo apt-get -f install
```

Generic .tar.gz Distribution

We also provide a generic .tar.gz package. Please review the [System Requirements and Supported Environments](#) topic to ensure compatibility. You will also need to install the **dependencies** described in the [Requirements](#) section. Once the dependencies are installed, you can install the **.tar.gz** with the following commands as root:

Viewer Example

```
tar -xzvf prizmdoc_client*.tar.gz
cd prizmdoc_client_*
ls prizm-*.tar.gz | xargs -n1 tar xzf
cp -R prizm /usr/share/
```

Server Example

```
tar -xzvf prizmdoc_server*.tar.gz
cd prizmdoc_server_*
ls prizm-*.tar.gz | xargs -n1 tar xzf
cp -R prizm /usr/share/
```

3. Add symbolic links to the fonts directory and update system fonts cache to enable the usage of installed fonts by PrizmDoc services.

Server Example

```
ln -s /usr/share/prizm/modules/poppler/fonts/accusoft_prizm_fonts.conf
/etc/fonts/conf.d/99-accusoft_prizm_fonts.conf fc-cache -f
```

Step 3 - Configure Your License

1. If you have a paid license, configure your license (see [Licensing](#) for more information). If you are evaluating the product, you can skip this step.

Step 4 - Verify that the Installation was Successful

1. **Start PrizmDoc Server:**

```
/usr/share/prizm/scripts/pccis.sh start
```

See [Starting and Stopping PrizmDoc Server on Linux](#) for more information.

2. **Wait for PrizmDoc Server to become healthy.** Poll `http://yourserver:18681/PCCIS/V1/Service/Current/Health` with a `GET` request until it returns `HTTP 200`, indicating the server is healthy. See the [Health Status API](#) for more information.

Once PrizmDoc Server reports healthy, your installation is complete.

Uninstall Traditional Linux Install Packages

NOTE: This topic applies to traditional Linux install packages only.

Introduction

To uninstall PrizmDoc from your Linux system, perform the following steps:

Make sure you log in as **root** to the machine.

1. Stop the service:

NOTE: This will depend on where the product is installed. The command for default installations will look like the following:

```
/usr/share/prizm/scripts/pccis.sh stop
```

2. Remove the installed files:

Ubuntu:

```
apt-get purge prizm-services.*
```

NOTE: This will remove all configuration files. If you would like to keep configuration files you

can instead run `apt-get remove prizm-services.*` however this will leave behind configuration files and may cause issues in the future. You may want to create a backup of configuration files before purging as an alternative.

Red Hat/CentOS:

```
yum remove prizm-services*
```

IMPORTANT: This will not properly execute if run in a directory with files matching the wildcard, for example, the `/usr/share/prizm/` directory or the directory where the downloaded Prizm services `.deb` or `.rpm` files are located.

Generic Package:

Remove symbolic links to the fonts directory and update system fonts cache.

```
rm /etc/fonts/conf.d/99-accusoft_prizm_fonts.conf  
fc-cache -f**
```

Remove PrizmDoc Server files.

```
rm -rf /usr/share/prizm/bin  
rm -rf /usr/share/prizm/conf  
rm -rf /usr/share/prizm/consul  
rm -rf /usr/share/prizm/java  
rm -rf /usr/share/prizm/libreoffice  
rm -rf /usr/share/prizm/libs  
rm -rf /usr/share/prizm/modules  
rm -rf /usr/share/prizm/mono  
rm -rf /usr/share/prizm/node.js  
rm -rf /usr/share/prizm/pccis  
rm -rf /usr/share/prizm/plu  
rm -rf /usr/share/prizm/schemas  
rm -rf /usr/share/prizm/scripts  
rm -rf /usr/share/prizm/services  
rm -rf /usr/share/prizm/src
```

NOTE: There may be temporary files left behind, like log and cache files, because they are not part of the installation packages. You may leave the temporary files or review them before deleting them.

Check PrizmDoc Server Health

Introduction

While running, a PrizmDoc Server instance is either considered healthy or unhealthy. When an instance reports as healthy, it is capable of handling new work normally. When an instance reports as unhealthy, that indicates something is wrong and may be unable to handle incoming work correctly.

Sometimes, an instance going unhealthy is temporary. After a brief period of time, the instance becomes healthy again. In these kinds of situations, there may be nothing you need to do as the administrator.

However, other times, an instance becomes stuck in an unhealthy state. When this happens, you will need to restart the PrizmDoc Server instance to restore it to a healthy state. For more details, refer to [Resolving PrizmDoc Server Health Issues for both Linux & Windows](#).

As an administrator of PrizmDoc Server instances, you should adopt an appropriate policy for monitoring instance health and restarting any instances which remain unhealthy for too long. The exact policy you adopt is up to you and will depend upon your infrastructure and workflow, but you might consider something like checking instance health once every 30 seconds and restarting any instance if it reports unhealthy two consecutive times.

Using the Health API

To check whether or not a server is healthy, simply send an HTTP request [GET /PCCIS/V1/Service/Current/Health](#). If the server is healthy, **HTTP 200** will be returned; if unhealthy, **HTTP 500** will be returned. Note that if PrizmDoc Viewer has just started, HTTP 500 may be returned for a short time until the system has completely started up.

Cluster Mode: Use the CEP

If you are running in cluster mode, send your request to the Cloud Entry Point (CEP), typically running on port 18681:

```
GET server:18681/PCCIS/V1/Service/Current/Health
```

Single-Server Mode: Use the SEP

If you are running in single-server mode, send your request to the Server Entry Point (SEP), typically running on port 18681:

```
GET server:18681/PCCIS/V1/Service/Current/Health
```

HTTP Response

- **HTTP 200** - Server is healthy and running normally.
- **HTTP 500** - Server is unhealthy.

Upgrade PrizmDoc Viewer

Introduction

This topic covers the details you need to upgrade PrizmDoc Viewer.

PrizmDoc Server Upgrade

Starting with version 13.14, PrizmDoc Server provides the ability to retain configuration settings when upgrading from a previous version.

PrizmDoc Server Upgrading Version 13.3 and Higher

Starting with version 13.14, when the installed version to upgrade is 13.3 or higher, the PrizmDoc Server preserves the server-side `prizm-services-config.yml` and `pcc.config` files.

PrizmDoc Server Upgrading a Version Prior to 13.3

IMPORTANT: Before starting, **make a backup** of the following configuration files so you can use them as a reference when re-configuring the new version after installation. This should be done **before you run the PrizmDoc Viewer installer**, as all configuration files will be replaced with new ones (resetting them to their default configuration):

- **Linux:**
 - PrizmDoc Server Configuration:
 - `/usr/share/prizm/prizm-services-config.yml`
 - `/usr/share/prizm/PCCIS/ServiceHost/pcc.config`
- **Windows:**
 - PrizmDoc Server Configuration:
 - `C:\Prizm\prizm-services-config.yml`
 - `C:\Prizm\PCCIS\ServiceHost\pcc.config`

PrizmDoc Application Services Upgrade

Starting with version 13.14, the PrizmDoc Application Services provide the ability to retain configuration settings when upgrading from a previous version.

PrizmDoc Application Services Upgrading Version 13.8 and Higher

Starting with version 13.14, when the installed version to upgrade is 13.8 or higher PrizmDoc Application Services are preserving `pcc.nix.yml` and `pcc.win.yml` files.

PrizmDoc Application Services Upgrading a Version Prior to 13.8

IMPORTANT: Before starting, **make a backup** of the following configuration file so you can use them as a reference when re-configuring the new version after installation. This should be done **before you run the PrizmDoc Application Services installer**, as configuration file will be replaced with new one (resetting them to their default configuration):

- **Linux:**
 - PrizmDoc Application Services Configuration:
 - `/usr/share/prizm/pas/pcc.nix.yml`
- **Windows:**
 - PrizmDoc Application Services Configuration:
 - `C:\Prizm\pas\pcc.win.yml`

How To Upgrade

1. Download the **latest version** of PrizmDoc Server and PrizmDoc Viewer client packages for your operating system from [PrizmDoc Viewer: Current Builds](#).

NOTE: On Linux, stop **PrizmDoc Server** before upgrade to prevent any possible side effects.

2. Install **PrizmDoc Server** first, and then the **PrizmDoc Application Services**. This will perform an in-place upgrade of the **PrizmDoc Server** and the **PrizmDoc Application Services** to the new version.

3. At the end of the server installation, the install may request a reboot.
4. Make a **backup** of your new configuration files as listed above.
5. Modify each of the **new configuration files** and make the same changes that you had in the older configuration files.

NOTE: Do not just replace the new configuration files with the older version of the configuration files, as new configurations may have been introduced in the new version and they would be lost.

6. Restart the **PrizmDoc Server** and **PrizmDoc Application Services** to ensure the newly configured file changes take effect.

NOTE: If either service fails to start with an error after modifying the configuration files, replace the configuration files with the original copy of the configuration files and try making the changes again.

7. If you are upgrading **PrizmDoc Application Services** from version 12.x and you are **using viewing packages and have an existing database** then you need to update your existing database by executing an additional script called `addTenantId.sql` to add a new field to one of the existing tables. The script is in the following location:

- **Linux:**
 - `/usr/share/prizm/pas/mssql-scripts/addTenantId.sql` (for MS SQL database)
 - `/usr/share/prizm/pas/mysql-scripts/addTenantId.sql` (for MySQL database)
- **Windows:**
 - `C:\Prizm\pas\db\mssql-scripts\addTenantId.sql` (for MS SQL database)
 - `C:\Prizm\pas\db\mysql-scripts\addTenantId.sql` (for MySQL database)

Follow your database management system's documentation to run the script. (For example, use the 'mysql' command line client for MySQL database and 'sqlcmd' for MS SQL.) If you need database connection parameters, refer to the `database` parameters in the **PrizmDoc Application Services** configuration file:

- **Linux:**
 - `/usr/share/prizm/pas/pcc.nix.yml`
- **Windows:**
 - `C:\Prizm\pas\pcc.win.yml`

Configure a Cluster

Introduction

Setting up multiple servers is not required for evaluation purposes.

However, if you are interested in installing and configuring PrizmDoc Server on multiple servers, this topic provides an overview with links to specific how-to instructions.

Setup Options

PrizmDoc Server default installation and configuration is designed with the intent to handle all requests and processing on a single server; however, running a single server will limit the bandwidth available for fulfilling requests. To address that problem, PrizmDoc Server can be installed on multiple servers and configured to route requests among them.

A cluster installation follows the same process as a single-server installation with some additional configuration steps once installation and licensing have been completed.

For an overview of how Cluster Mode works and steps to configure your server for cluster use, please see our [Cluster Mode](#) introduction. If you're an existing customer and already familiar with Cluster Mode, you can find information on our [Cluster Management API](#) here.

With a cluster configuration, there are special considerations for optimizing PrizmDoc Server's performance by configuring viewing sessions to use cached content. To learn more about relying on PrizmDoc Server's caching and to configure its use, see our topic on [Optimizing Cache Performance](#).

Some of the APIs also require special consideration when used with Cluster Mode. In Cluster Mode, each server handles a request from start to finish. For that reason, some requests (for example Work File, Markup Burner, and Content Conversion requests) will require an Affinity Token. For more information on Affinity Tokens, and the steps required to use those APIs in Cluster Mode, please see our [Affinity Tokens & Cluster Mode](#) topic.

Licensing

License Types

Metered License

A **Metered License** allows you to use all features of the product on as many servers as you want as long as your license is current. As you use the product, the number of documents processed will be automatically reported to Accusoft. When you renew your license, the new expiration date is applied automatically.

To purchase a Metered License, contact info@accusoft.com.

OEM License

An **OEM License** allows you to use [the features you have paid for](#) according to the terms of your agreement until your license expiration date. All usage reporting is manual, and no network requests are made to Accusoft. When you renew your license, you will be given a new license key with an updated expiration date, and you will be required to reconfigure and redeploy your PrizmDoc Server instances with your new license key.

To purchase an OEM License, contact info@accusoft.com.

Cloud License (Deprecated)

A **Cloud License** allows you to use [the features you have paid for](#) on a limited number of logical CPU cores until your license expiration date. At runtime, the total number of server virtual CPU cores is tracked and, if bringing up a server exceeds the core limit, its functionality is disabled. In order to track the number of cores in use, a Cloud License requires you to configure an AWS S3 bucket which all of your PrizmDoc Server instances can read and write from.

Despite the name, a Cloud License can be used for both on-premise and cloud deployments as long as your PrizmDoc Server instances have access to an S3 bucket which you own. But a Cloud License has several disadvantages: 1) you must correctly configure an AWS S3 bucket and properly configure your PrizmDoc Server instances to access it, 2) renewing your license requires reconfiguration and redeployment, and 3) you cannot scale beyond the number of virtual CPU cores you paid for without upgrading to a new license.

The Cloud License type has been deprecated, and the ability to run the product with a Cloud License will be removed in a future release. If you are currently using a Cloud License, we recommend you migrate to a new Metered License.

To purchase a new Metered License or renew your existing Cloud License, contact info@accusoft.com.

Node-Locked License (Deprecated)

A **Node-Locked License** allows you to use [the features you have paid for](#) on a specific number of physical machines ("nodes") which your license has been "locked" to. Because this license type is locked to physical machines, installing the license is not as simple as configuring the product with a particular license key. Instead, you must run a tool (the Prizm Licensing Utility) which communicates with Accusoft to register the physical machine with your license.

A Node-Locked License is not suitable for use with virtual machines, Docker containers, or the cloud.

The Node-Locked License type has been deprecated, and the ability to run the product with a Node-Locked License will be removed in a future release. If you are still using a Node-Locked License, we recommend you migrate to a new Metered License.

To purchase a new Metered License or renew your existing Node-Locked License, contact info@accusoft.com.

Feature Licensing

For [OEM Licenses](#), [Cloud Licenses](#), and [Node-Locked Licenses](#), you can choose whether or not you want to pay to activate the following features:

- **Microsoft Office Conversion (MSO)** - Native rendering of Word, Excel, and PowerPoint documents when Microsoft Office is installed on the machine where PrizmDoc Server is running. For configuration, see:
 - [Working with PrizmDoc Viewer > Developer Guide > PrizmDoc Server > How To > Natively Render Microsoft Office Documents](#)
- **Form Field Detection** - Enables APIs for automatically detecting form fields in PDFs and images. See:
 - [PAS API > Developer Reference > Form Extractors](#)
 - [PrizmDoc Server API > Developer Reference > Form Extractors](#)
 - [E-Signature API > Module: form-extraction](#)
- **OCR** - Enables Content Conversion Service option to automatically recognize text in raster documents (such as image-only PDFs or TIFFs) to produce a text-searchable PDF. See:
 - [API Reference > PrizmDoc Server API > Content Conversion Service](#)

For [Metered Licenses](#), there are never any restrictions on features; all product features are always enabled.

If you need to purchase a feature which your current license does not enable, contact info@accusoft.com.

Metered License

Overview

A **Metered License** is an internet-connected license which allows you to use all features of the product on as many servers as you want as long as your license is current and your PrizmDoc Server instances are able to periodically contact Accusoft.

At runtime, each PrizmDoc Server instance will periodically contact Accusoft to 1) report the number of documents you have processed and 2) validate your license is still current. See the FAQ below for more information.

To purchase a Metered License, contact info@accusoft.com.

Requirements

- You must be running PrizmDoc Server v13.15 or greater.
- Your PrizmDoc Server instances must be able to send HTTP requests to Accusoft (<https://license.accusoft.com>) during startup and periodically while running (see the FAQ below for more information).

Usage

To configure a PrizmDoc Server instance to use a Metered License:

1. Update two values in your [Central Configuration](#) file (`prizm-services-config.yml`):
 - `license.key` - Set to your Metered License key (like `abc...`).
 - `license.solutionName` - Set to any value of your choice. The value does not matter, but a value must be set or the product will not start.
2. [Start/Restart PrizmDoc Server](#) to apply the new license key.

Once PrizmDoc Server starts and [reports healthy](#), you are licensed and ready for production traffic.

FAQ

What counts as a "processed document"?

We count *the number of times any document is processed*.

A *process* is anything that does work on a document, such as a viewing session, a markup burning operation, or a content conversion operation. For example:

- Creating a viewing session for one document counts as 1 processed document.
- Creating a viewing session to compare two DOCX files counts as 2 processed documents.
- Creating a viewing session for an email with attachments will count as 1 processed document for the email itself and 1 processed document for each top-level attachment.
- Creating redaction definitions by looking for text in a document by regex patterns counts as 1 processed document.
- Burning redactions or annotation definitions into a document counts as 1 processed document.
- Using the Content Conversion Service to convert or combine documents will count as ***n*** processed documents based on the number of **input** documents you use. For example, if you convert a DOCX to a PDF, there is only one input document, so this will count as 1 processed document. If you convert a PDF to multiple PNG files, one output per page, there is still only one input document, so this will count as 1 processed document. If you combine three files into a single output, there are three input documents, so this will count as 3 processed documents.

If the same document is processed repeatedly, we will count each time it is processed. For example, if a single document is viewed 25 times, it will count as 25 processed documents.

NOTE: You can view your usage statistics for processed documents in the Usage tab of the [Accusoft Customer Portal](#).

What data is reported back to Accusoft?

We report the number of processed documents for various periods of time (**typically 4 hour intervals**).

A report back to Accusoft will include a list of time periods. Each reported time period will include:

- Timestamp of when the period began
- Duration of the period

- Count of documents processed
- An auto-generated id

No other data is reported back to Accusoft.

How often does PrizmDoc Server contact Accusoft at runtime?

By default, PrizmDoc Server instances will contact Accusoft **once every 18 hours**. The exact interval used by your license is subject to change by Accusoft.

What happens if PrizmDoc Server is unable to reach Accusoft?

When starting, PrizmDoc Server MUST be able to reach Accusoft to validate your license is current. If it cannot, it will refuse to start.

While running, PrizmDoc Server will gracefully allow a temporary disruption in connectivity to Accusoft. However, if PrizmDoc Server is unable to reach Accusoft for a long period of time (**6 hours** by default), it will disable product functionality and cause all REST API requests to return `HTTP 580 ProductNotLicensed`. Once product functionality is disabled, you must restart PrizmDoc Server for functionality to be restored.

OEM License

Overview

An **OEM License** allows you to use [the features you have paid for](#) according to the terms of your agreement until your license expiration date.

All usage reporting is manual, and no network requests are made to Accusoft.

Whenever you renew an OEM License, you will be given a new license key with an updated expiration date, and you will be required to reconfigure and redeploy your PrizmDoc Server instances with the new license key.

To purchase an OEM License, contact info@accusoft.com.

Usage

To configure a PrizmDoc Server instance to use an OEM License:

1. Update two values in your [Central Configuration](#) file (`prizm-services-config.yml`):
 - `license.key` - Set to your OEM License key (like `2.0.abc...`).
 - `license.solutionName` - Set to your solution name (like `My Product`), chosen at the time of purchase. This value must match the value encoded in the license key itself.
2. [Start/Restart PrizmDoc Server](#) to apply the new license key.

That's it. Once PrizmDoc Server starts and [reports healthy](#), you are licensed and ready for production traffic.

Cloud License (Deprecated)

Overview

A **Cloud License** allows you to use [the features you have paid for](#) PrizmDoc Server instances **across a paid number of total logical CPU cores** until your license expiration date. At runtime, the total number of logical CPU cores is tracked and, if

bringing up a server exceeds the core limit, its functionality is disabled. In order to track the total number of logical cores in use, a Cloud License requires you to configure an AWS S3 bucket which all of your PrizmDoc Server instances can read and write from (see requirements below).

Despite the name, a Cloud License can be used for both on-premise and cloud deployments as long as your PrizmDoc Server instances have access to an S3 bucket which you own. However, a Cloud License has some disadvantages: 1) you must correctly configure an AWS S3 bucket and properly configure your PrizmDoc Server instances to access it, 2) renewing your license requires reconfiguration and redeployment, and 3) you cannot scale beyond the number of virtual CPU cores you paid for without upgrading to a new license.

Deprecation Notice

The Cloud License type has been deprecated, and the ability to run the product with a Cloud License will be removed in a future release. If you are currently using a Cloud License, we recommend you migrate to a new [Metered License](#).

Requirements

NOTE: PrizmDoc Cloud Licensing does not support Docker orchestration systems, such as AWS Fargate/ECS/EKS, Google Kubernetes, Azure Container Instances, etc.

S3 Bucket with Appropriate Permissions Configured

To use a Cloud License, you need to provide an S3 bucket to PrizmDoc with read, write, and delete permissions:

listObjects

- [AWS S3 listObjects docs](#)
- Requires [s3:ListBucket](#)

putObject

- [AWS S3 putObject docs](#)
- Requires [s3:PutObject](#)

deleteObjects

- [AWS S3 deleteObjects docs](#)
- Requires [s3:DeleteObject](#)

At runtime, each of the PrizmDoc Server instances in your cluster will read and write extremely-small, temporary files into this bucket as part of licensing enforcement.

Usage

To set up a Cloud License, you will need to do the following:

1. Set up an **S3 bucket**.
2. Purchase a **Cloud License** from Accusoft.
3. Get your **license key** from the [Accusoft customer portal](#).
4. Configure your **PrizmDoc Server instances** with the necessary **AWS credentials**.
5. Configure your **PrizmDoc Server instances** with your **license key**.

The sections below provide additional information for each step.

1 - Set up an S3 bucket

First, you must have an [Amazon AWS account](#) in order to create an S3 bucket. Log into your AWS account console and create an [S3 bucket](#) which PrizmDoc can read and write to at runtime. See the [Amazon S3 documentation](#) for more information on setting up an S3 bucket and acquiring credentials.

NOTE: We recommend that you create a **dedicated** AWS user and bucket with read/write credentials.

2 - Purchase a Cloud License from Accusoft

To purchase a Cloud License for PrizmDoc Viewer, contact info@accusoft.com.

During purchase, specify the maximum number of logical cores to be provided by the license. Make sure to account for production, support, and any ongoing development usage; you may want to consider purchasing separate licenses for both production and development.

3 - Get Your License Key from the Accusoft Customer Portal

Once purchased, visit the [Accusoft Customer Portal](#) to view your purchased license and provide the S3 bucket name that you will use for Cloud Licensing. After providing your S3 bucket name, your Cloud License key will be given to you. > **NOTE:** It is your responsibility to provide a valid S3 bucket and to ensure that you have the proper credentials to access that bucket.

1. Enter your **Amazon S3 bucket** name in the field provided and click **Activate License**:

Deployment Cloud Annual

Version 13

PrizmDoc 13 Cloud Annu

Quantity: 1
Expiration: Mar 15, 2020

download license

license

Cloud Annual Licensing Instructions

This Cloud License should be provided to the Prizm Licensing Utility installed with Prizm Content Connect. Please see the Cloud Licensing section of the product help file, which can be accessed via a shortcut in the start menu or found [online](#). The help file provides instructions on how to apply a license using the Prizm Licensing Utility and how to apply the license directly to the properties files.

Solution Name: TestPrizmCloudAnnual

Cloud Provider Type: s3
Amazon S3 Bucket Name:

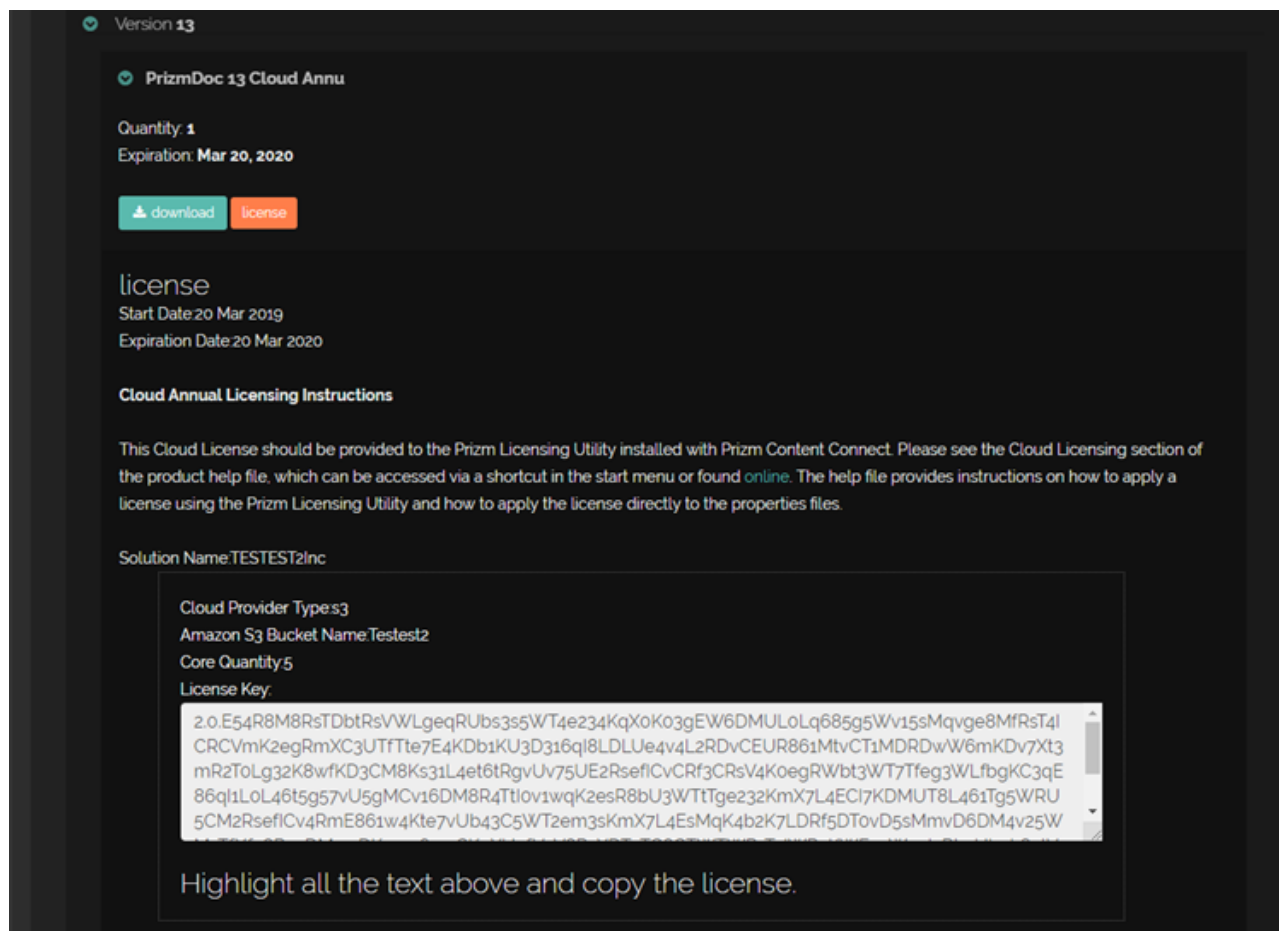
Core Quantity: 1
activate license

NOTE: You **must** enter your S3 bucket name correctly. Make sure there are no leading or trailing spaces! If you make a mistake, your license key will be unusable.

2. Select and copy **all of the text** in the License Key field below. This is your activated license key and you will need to **save it** for later use when configuring PrizmDoc Viewer on your server:

Prizm

Deployment Cloud Annual



3. Click **Download** to go to the product download page and download your product.
4. Continue with **Section 4** below.

4 - Configure Your PrizmDoc Server Instances with the Necessary AWS Credentials

When running PrizmDoc with a Cloud License on your server, it will require access to the Amazon S3 bucket you provided when your license was generated. It is important to note that PrizmDoc Server itself requires no knowledge of your S3 credentials. PrizmDoc Server accesses Amazon S3 assuming your credentials have been provided using one of the methods defined in the [Amazon SDK documentation](#) that do not require explicitly providing them to our product. As shown below, using method 1, 2, or 3 will provide access to AWS S3.

If you are new to Amazon Web Services (AWS), you should familiarize yourself with the [Security Credentials](#) to get an overview.

There are three options for configuring your AWS credentials for Cloud Licensing. They are listed in the order of recommendation:

1. Configure **IAM roles for Amazon EC2** (if running on EC2), or
2. Configure the **Shared Credentials File** (~/.aws/credentials), or
3. Configure **Environment Variables**.

NOTE: When setting credentials, be sure that the user under which the PrizmDoc service is running is the same user for which you are configuring the credentials. For example, consider the case where you configure the credentials for your personal user, but PrizmDoc is running as the root user. In this case the service will start, but then shut down shortly after because the root user will be unable to access the S3 bucket specified in your Cloud License key.*

Examples using IAM Roles are beyond the scope of this topic, but if your cluster is running on AWS EC2 instances, you can find information about IAM Roles and their configuration in the [Amazon IAM documentation](#).

An example using method 2 above, Shared Credentials Files, are shown below. You can configure a shared credentials file for the user under which PrizmDoc will execute:

In both Linux and Windows, the credentials file content is the same.

Example Credentials File Content

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

Credential files are stored under an `.aws` directory for the user under which PrizmDoc is executing. The credential file paths below are for an example called: `prizmdocuser`.

In Linux

```
/home/prizmdocuser/.aws/credentials
```

NOTE: If your PrizmDoc Server is configured to start/stop with the system, you must modify the configuration section of `/usr/share/prizm/pccis.sh` script: set the `AWS_HOME` variable to the path of the directory which contains the AWS folder (`.aws`). For the example above, it will be `AWS_HOME=/home/prizmdocuser/_`

When using Docker

Map the `.aws` directory to prizmdoc container's file system when starting the container:

```
docker run --rm --env ACCEPT_EULA=YES --publish 18681:18681 --volume
$(pwd)/config:/config --volume $(pwd)/logs:/logs --volume $(pwd)/data:/data --volume
$(pwd)/.aws:/root/.aws --name prizmdoc-server accusoft/prizmdoc-server
```

In Windows

```
C:\Users\prizmdocuser\.aws\credentials
```

Examples using method 3 above, Environment Variables, are shown below. You can configure your server to export the credentials to the environment for the user under which PrizmDoc will execute:

In Linux

Exporting environment variables under Linux is usually done in the user's `.profile` file in their home directory (e.g. for a user named `prizmdocuser` these would be added to `/home/prizmdocuser/.profile`). > **NOTE:** The following environment variables need to be configured on the Docker container since the licensing component in the app will use them to access the S3 bucket.

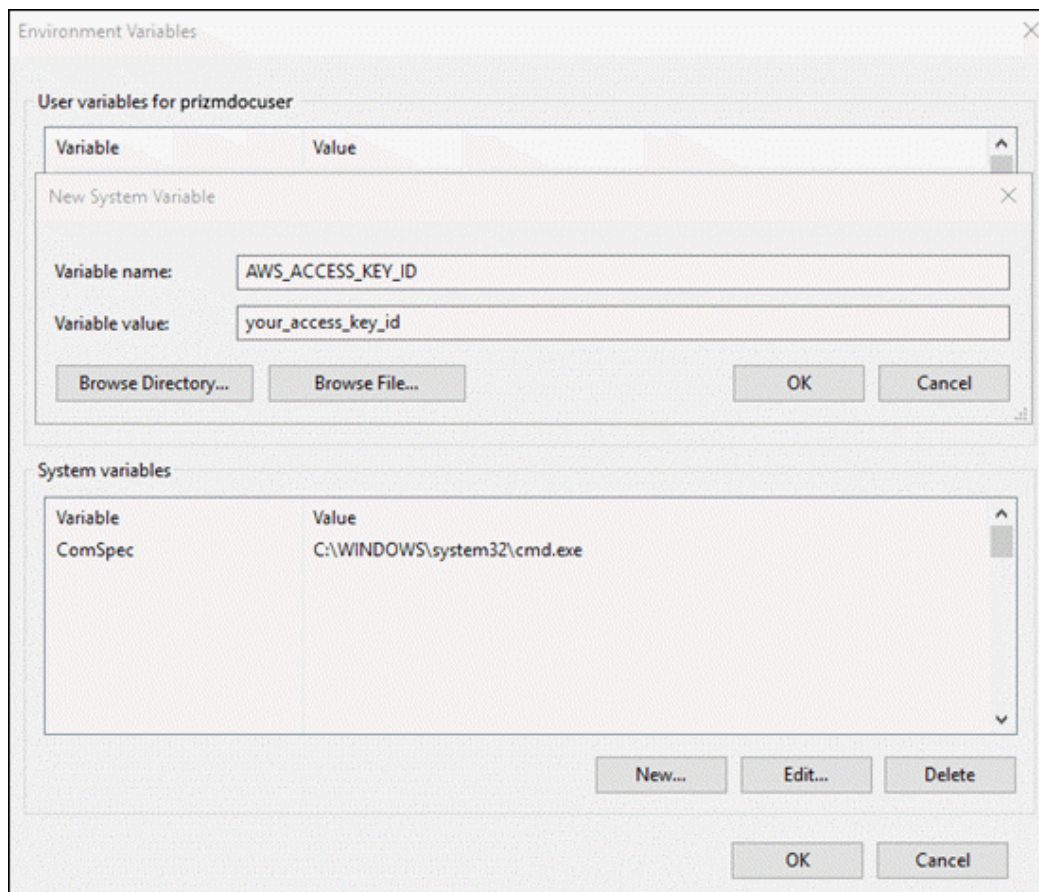
Example

```
export AWS_ACCESS_KEY_ID=your_access_key_id
export AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

In Windows

Exporting environment variables under Windows is done in the Environment Variables control panel under the System Properties.

Example



5 - Configure Your PrizmDoc Server Instances with Your License Key

To configure a PrizmDoc Server instance to use a Cloud License key:

1. Update two values in your [Central Configuration](#) file (`prizm-services-config.yml`):
 - o `license.key` - Set to your OEM License key (like `2.0.abc...`).
 - o `license.solutionName` - Set to your solution name (like `My Product`), chosen at the time of purchase. This value must match the value encoded in the license key itself.
2. [Start/Restart PrizmDoc Server](#) to apply the new license key.

Once PrizmDoc Server starts and [reports healthy](#), you are licensed and ready for production traffic.

Limiting the Number of CPU Cores Used by Docker Containers

If you run multiple `prizmdoc-server` containers on the same host, or want to limit the CPU consumption for `prizmdoc-server` containers, you should use the CPU affinity mask when starting the container. This allows PrizmDoc to correctly count the logical CPU cores used by the container, as opposed to the total number of host's CPU cores. Use Docker `run` command

parameter `--cpuset-cpus` to specify the affinity mask.

Example

This example assigns cores 0 and 1 for running the container:

```
docker run --rm --env ACCEPT_EULA=YES --publish 18681:18681 --volume  
$(pwd)/config:/config --volume $(pwd)/logs:/logs --volume $(pwd)/data:/data --volume  
$(pwd)/.aws:/root/.aws --name prizmdoc-server --cpuset-cpus 0-1 accusoft/prizmdoc-server
```

Node-Locked License (Deprecated)

Overview

A **Node-Locked License** allows you to use [the features you have paid for](#) on a specific number of physical machines (“nodes”) which your license has been “locked” to. Because this license type is locked to physical machines, installing the license is not as simple as configuring the product with a particular license key. Instead, you must run a tool (the Prizm Licensing Utility, or PLU) which communicates with Accusoft to register the physical machine with your license.

A Node-Locked License is not suitable for use with virtual machines, Docker containers, or the cloud.

Deprecation Notice

The Node-Locked License type has been deprecated, and the ability to run the product with a Node-Locked License will be removed in a future release. If you are currently using a Node-Locked License, we recommend you migrate to a new [Metered License](#).

Usage

Using the GUI

On a Machine with Internet Access (GUI)

1. Obtain Licensing Information

When you signed up to evaluate PrizmDoc Viewer, you received an email with your Software Registration Information. The email message will contain a link to the Customer Portal, where you can obtain your licensing information for PrizmDoc Viewer. You will need to obtain the following information from the **Customer Licensing Portal**:

- **Solution Name** - This string is used to validate the configuration file.
- **Configuration File** - This file contains the information the Prizm Licensing Utility (PLU) requires in order to obtain a license for your system.
- **Access Key** - If you purchased several licenses, the Access Key is used to denote a *specific license* within the “pool” of purchased licenses. If this value is not provided, the Prizm Licensing Utility (PLU) will use the next license in your pool of licenses. If you purchased an *Annual License of PrizmDoc Viewer*, it is required that you *supply an Access Key*. If you have multiple licenses that expire on different days, the Access Key will allow you to differentiate between the licenses and ensure that the product will expire on the expected date.

2. Download and Install the Product

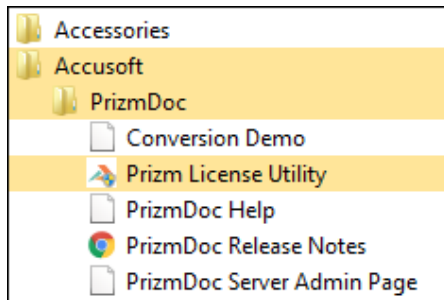
NOTE: *If you have already installed the product for evaluation or other purposes you can skip this step.*

The PrizmDoc Viewer product can be downloaded from the Accusoft web site: [PrizmDoc Viewer](#). After you have

downloaded the appropriate product installer, run the installer according the version's installation instructions.

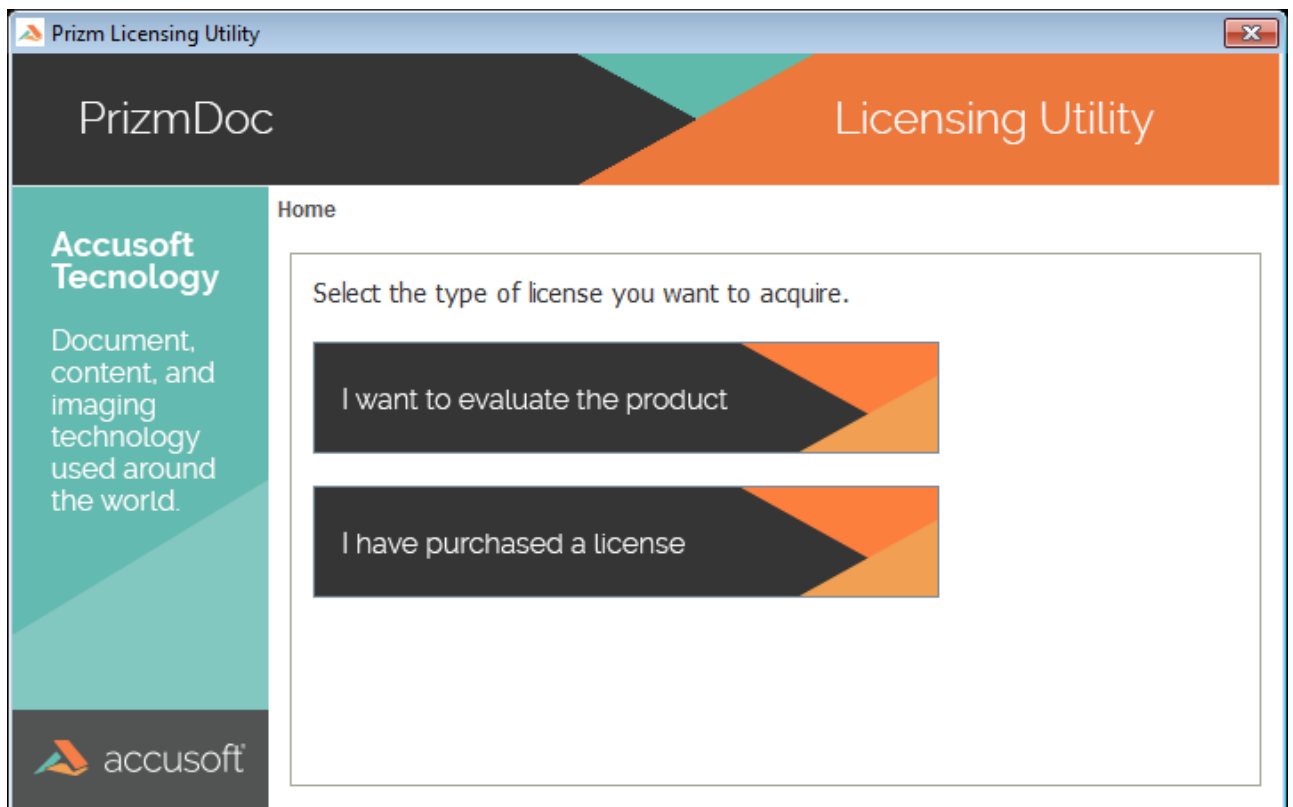
3. Run the Prizm Licensing Utility (PLU)

You can launch the utility from the following location:



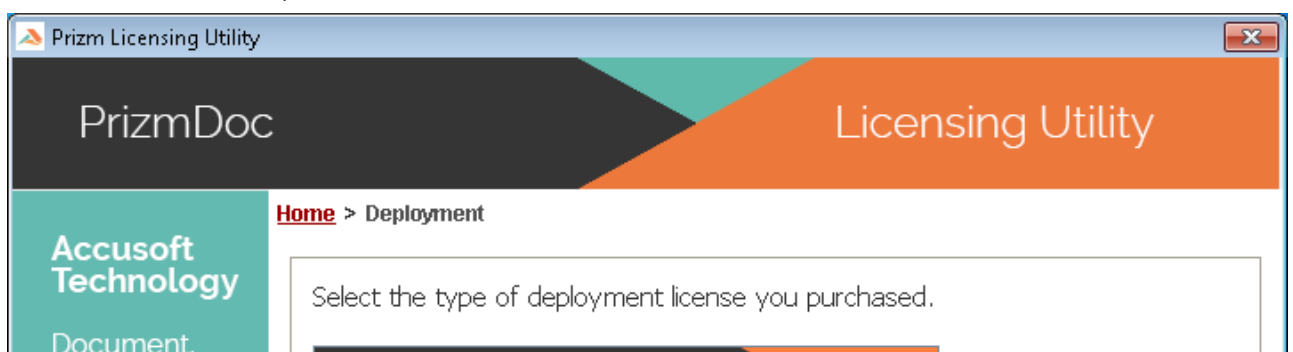
4. Select Deployment Option

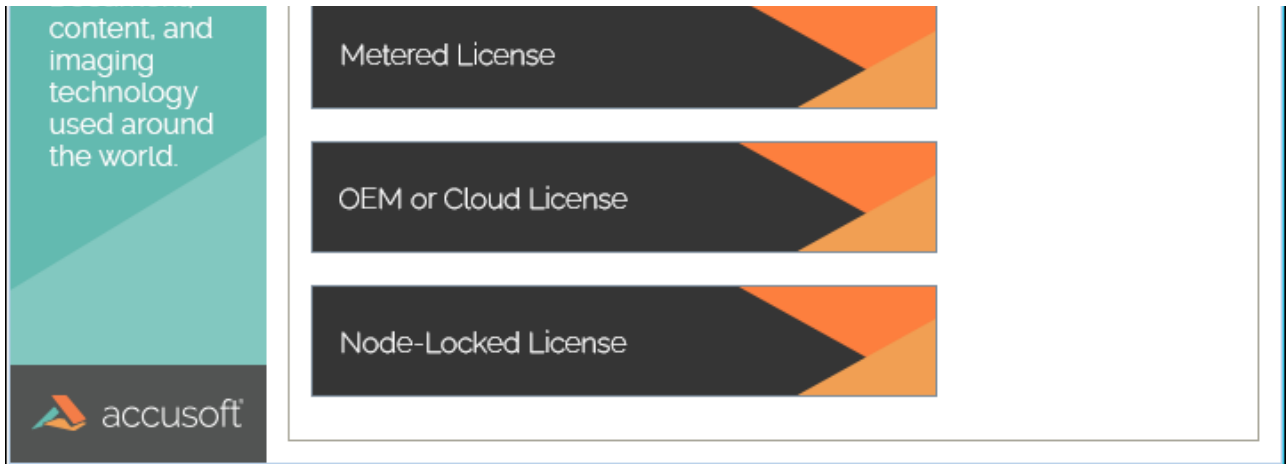
The Prizm Licensing Utility (PLU) provides options for both Evaluation and obtaining Deployment licensing. For this walk-through, we're using Deployment Licensing, so click the **I have purchased a license** button:



5. Select Node-Locked Option

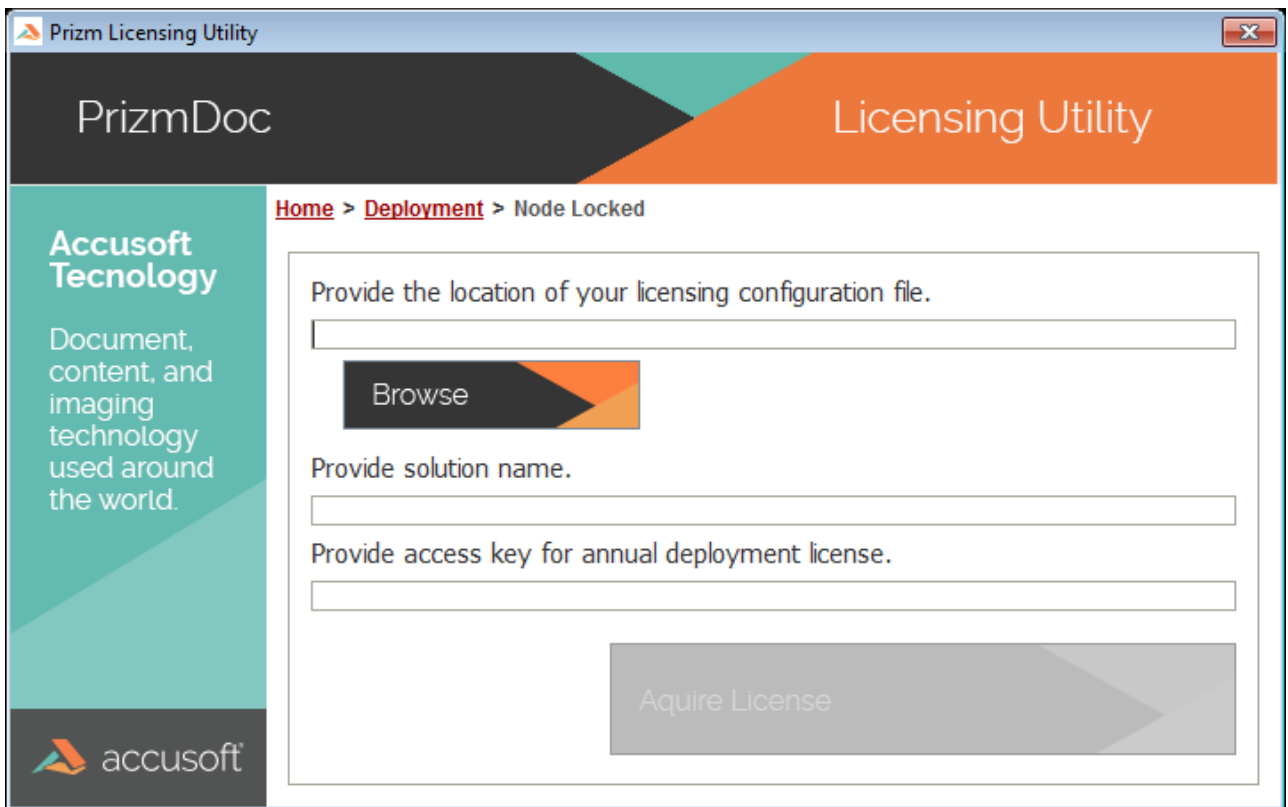
Click the **Node-Locked** option:





6. Provide Configuration File

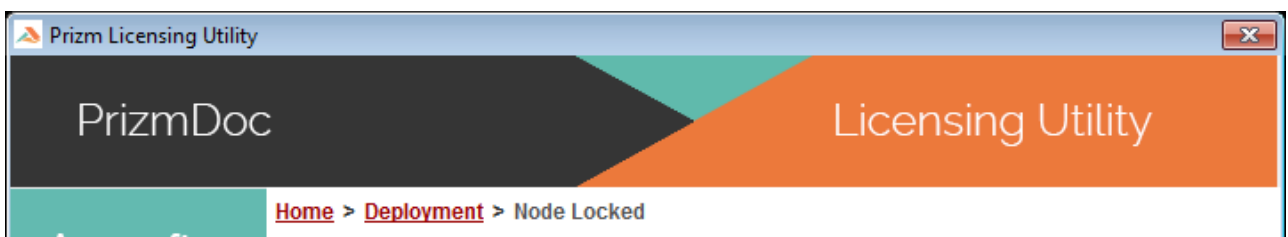
In Step 1 above, you acquired a "**configuration file**". Specify the path to the file directly using the text box, or browse to the file using the **Browse** option:



7. Enter Solution Name

In Step 1 above, you acquired the **Solution Name**. Enter the value that was assigned to your distribution into the text box below:

NOTE: This field is case-sensitive.



Accusoft Tecnology
Document, content, and imaging technology used around the world.

Provide the location of your licensing configuration file.

Browse

Provide solution name.

Provide access key for annual deployment license.

Aquire License

8. Enter your Access Key

The use of an Access Key is required for Annual Licenses. If you have multiple licenses that expire on different days, the Access Key will allow you to differentiate between the licenses and ensure that the product will expire on the expected date.

NOTE: The Access Key is a unique identifier for each distribution license purchased and may be required for your deployment.

For Annual Licenses, the use of an Access Key is required. If you have multiple licenses that expire on different days, the Access Key will allow you to differentiate between the licenses and ensure that the product will expire on the expected date. If an Access Key is not provided for an Annual License, the Prizm Licensing Utility (PLU) will use the next license available in your licensing "pool".

For Perpetual Licenses, the use of an Access Key is not required.

PrizmDoc Licensing Utility

Home > Deployment > Node Locked

Accusoft Tecnology
Document, content, and imaging technology used around the world.

Provide the location of your licensing configuration file.

Browse

Provide solution name.

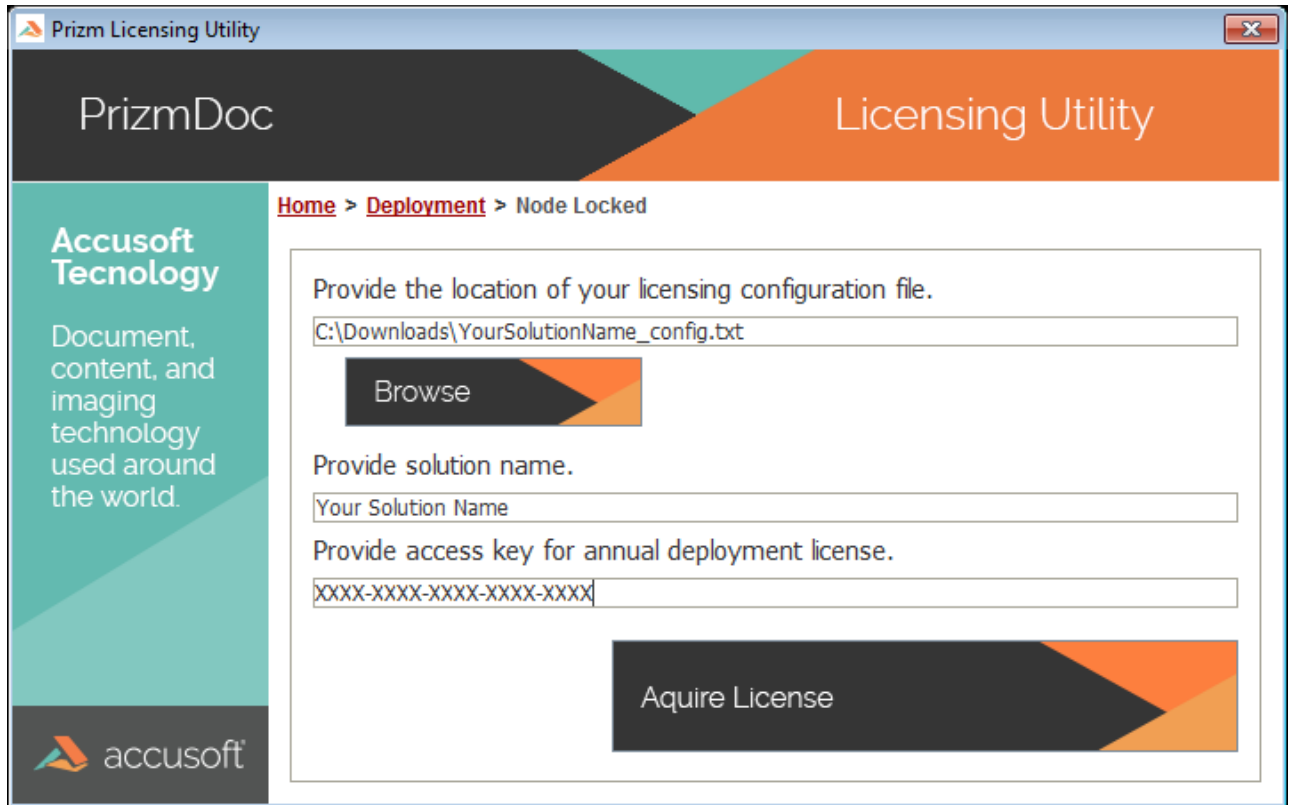
Provide access key for annual deployment license.

Aquire License

9. Acquire a License

Once all of the information has been provided, click the **Acquire License** button to register your system with Accusoft.

NOTE: If your system does not have Internet access to reach the Accusoft Licensing Services, then the Prizm Licensing Utility (PLU) will fail to register your system. In this case, you will need to follow the manual registration instructions below under the section, **On a Machine without Internet Access (GUI)**.



10. Registration Complete

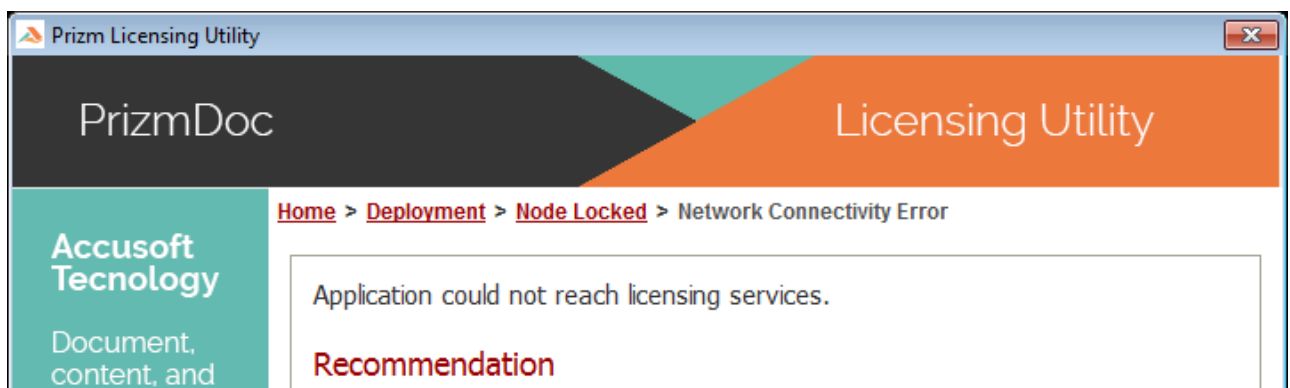
Your system has been licensed for use. If you purchased an Annual License, the Prizm Licensing Utility (PLU) will display the expiration date for the license which was acquired.

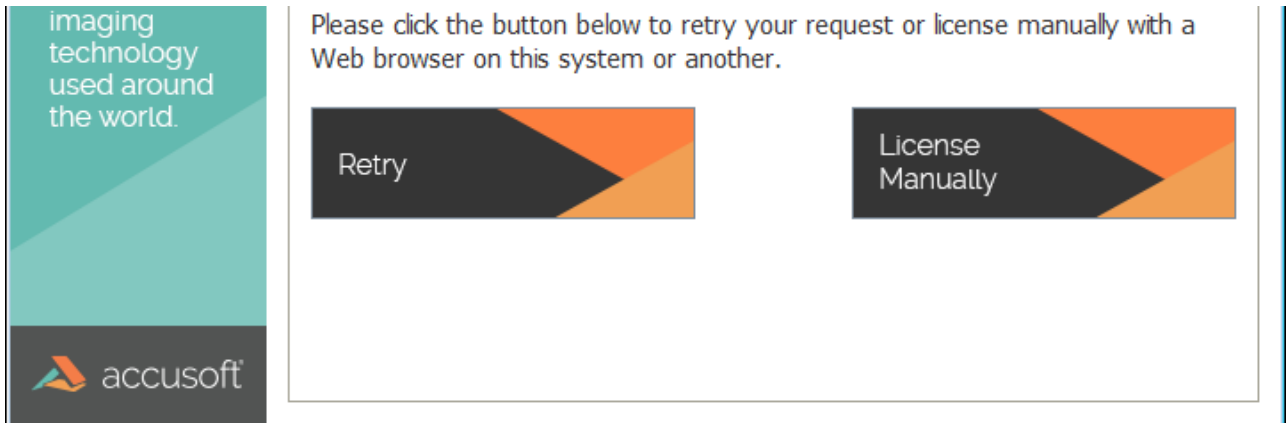
11. Restart PrizmDoc Server

Restart PrizmDoc Server for the licensing changes to take effect. See [Start & Stop PrizmDoc Server](#) for instructions.

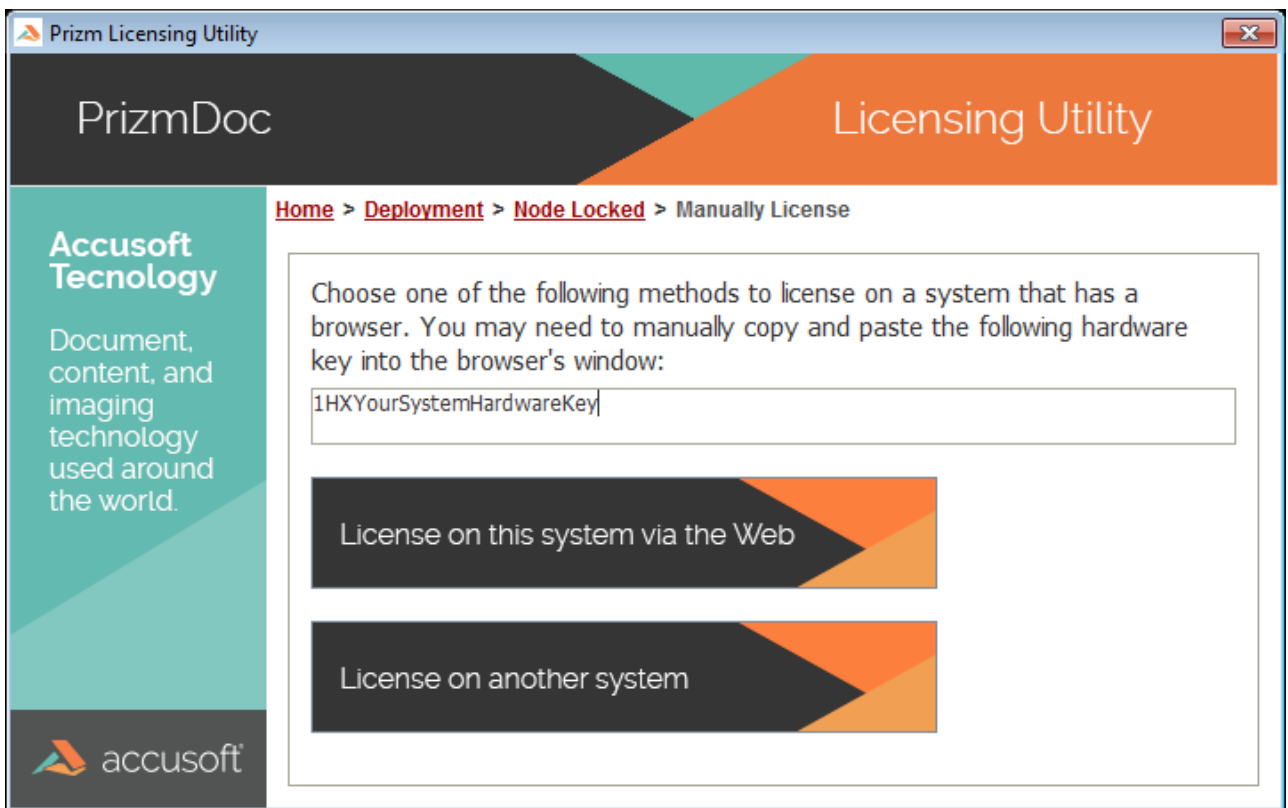
On a Machine without Internet Access (GUI)

1. If the Prizm Licensing Utility (PLU) is not able to contact the Accusoft Licensing Services, a dialog box will display stating that the "application could not reach the licensing services". You will have the option to retry the registration or to "License Manually". Select the **License Manually** option to proceed:





2. The Manually License dialog box will display a text box with your system **Hardware Key**. This key is used to identify your system during the registration process. This key will need to be supplied to the Accusoft Licensing Center in order to obtain a license to register the system. Using your mouse or keyboard select all of the text within the text box and copy it to the clipboard. (The shortcut keys of Ctrl+C will copy the selected data to the clipboard.)



3. Next, you will need to go to the Accusoft Licensing Center to obtain your license. The URL to this website is provided by the Prizm Licensing Utility (PLU), <https://licensing.accusoft.com/v1/WebDeployUser/WebDeployUser.aspx>. You have two options for getting this URL:

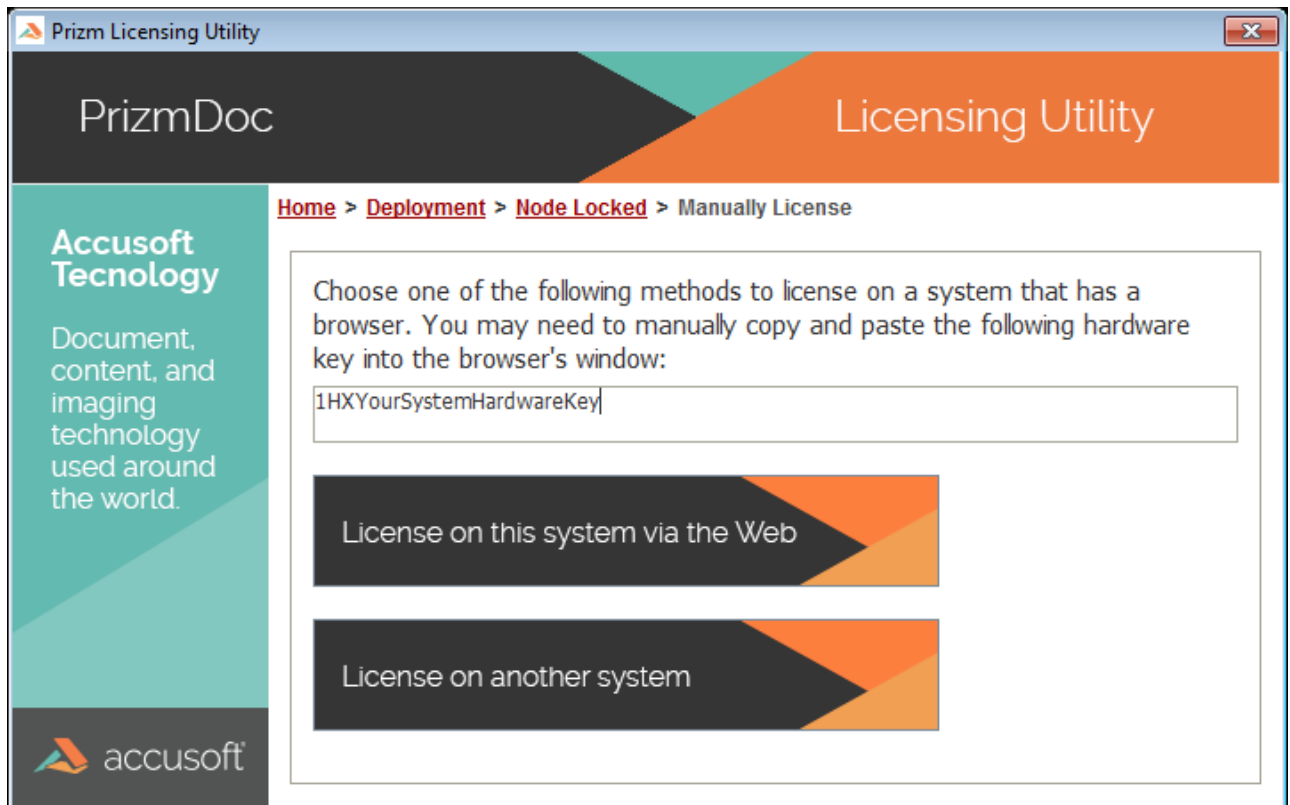
License on this system via the Web

Choosing this option will open the default web browser on your machine and navigate directly to the Accusoft Licensing Center. This option is recommended if, for example, your organization allows access to the public Internet only within the web browser through the use of proxy servers.

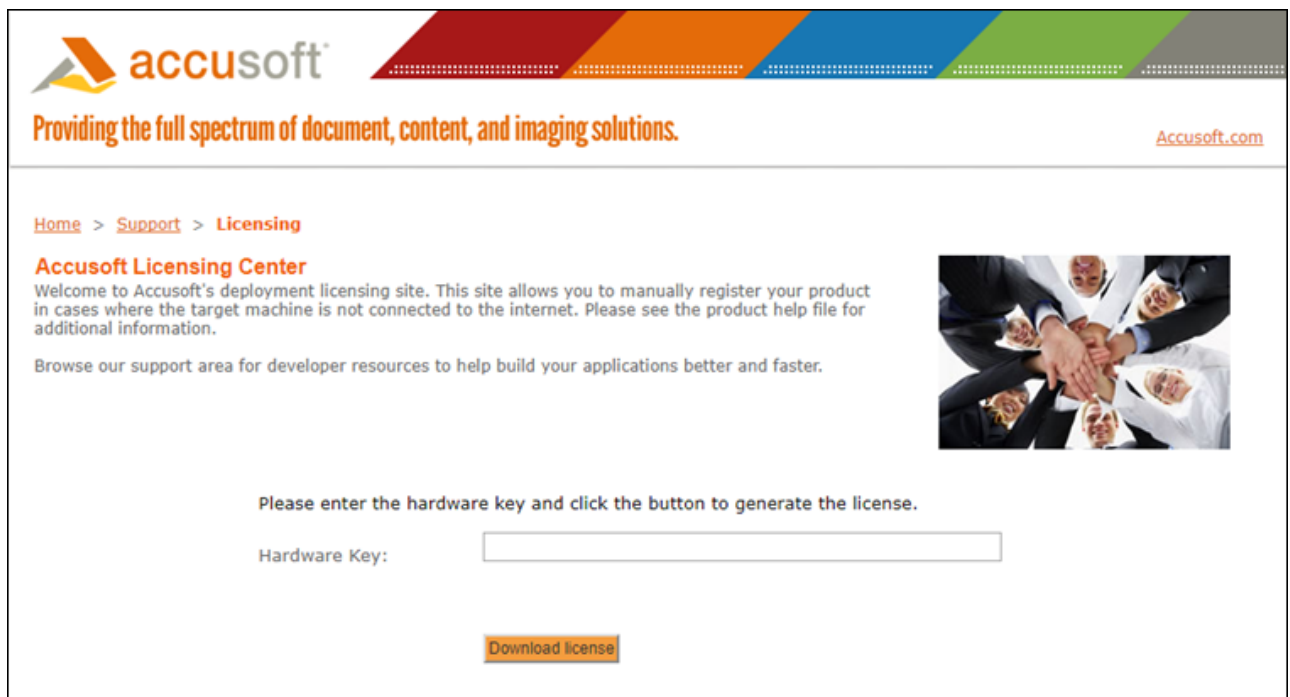
License on another system

Choosing this option will create an Internet Shortcut file (.URL). This is a simple text file that contains the full URL to the Accusoft Licensing Center website. In Windows environments, these files can be double-clicked to open the default web browser and navigate directly to the Accusoft Licensing Center website. If this action does not work in your environment, simply open the file in a text editor, copy the URL, and paste it into the address bar of your web

browser. This option is recommended if the system being registered does not have any connection to the Internet.



4. Once you have navigated your web browser to the Accusoft Licensing Center, you will need to paste your Hardware Key into the text box labeled **Hardware Key** (use the shortcut keys Ctrl + V to paste the Hardware Key):



5. Click **Download License** to have the Accusoft Licensing Center generate a license for your system. The license will be created and sent to your system as a text file.
6. Enter the License into the Prizm Licensing Utility (PLU) by copying and pasting the license information into the text box. Click **Apply License** to apply the License on the current system.

NOTE: If the Prizm Licensing Utility (PLU) was closed after you left it to go to the Accusoft Licensing Center, restart the application and perform all the previous steps to return to this screen in the Prizm Licensing Utility. You do not need to repeat the steps on the Accusoft Licensing Center web site.

Using the Command Line

In the steps above, we used the Prizm Licensing Utility (PLU) UI. However, the PLU also supports performing the same steps from the command line.

NOTE: If you have an updated license, you must re-start PAS and PrizmDoc Server in order to use the new license.

On a Machine with Internet Access (Command Line)

Usage

```
deploy get <configuration file> <solution name> \[<access key> outputUrl\]
```

Parameters

- **<configuration file>** – Path to the deployment configuration file. **Required.**
- **<solution name>** – Solution name for deployment licensing. **Required.**
- **<access key>** – Access key for annual deployment licensing. **Required.**
- **<outputUrl>** – A flag to output URL that can be used for licensing through the web portal in case of connectivity error. Can also be used if you are offline or have no Internet access. *Optional.*

Examples

The following example demonstrates obtaining and installing a deployment license:

```
java.exe -jar plu.jar deploy get "C:\Path to\YourSolutionName_Config.txt" "Your Solution Name"
```

The following example demonstrates obtaining and installing a deployment license with the access key:

```
java.exe -jar plu.jar deploy get "C:\Path to\YourSolutionName_Config.txt" "Your Solution Name" Your-Access-Key
```

The following example demonstrates obtaining and installing a deployment license with error handling to automatically output a URL to be used for licensing through the web portal. The URL should be opened from a browser on a computer with network connectivity. The URL will provide a license key:

```
java.exe -jar plu.jar deploy get "C:\Path to\YourSolutionName_Config.txt" "Your Solution Name" outputUrl
```

On a Machine without Internet Access (Command Line)

If you are licensing a server that does not have access to the internet, you can use the following example to set it up.

1. On the machine that does not have access to the internet, enter the following command:

```
java.exe -jar plu.jar deploy get "C:\Path to\YourSolutionName_Config.txt" "Your Solution Name" Your-Access-Key outputUrl
```

The `outputUrl` is shown below highlighted in white:

```
C:\Prizm\plu>java.exe -jar plu.jar deploy get "C:\Path to\Your_Config.txt" "YourSolutionName" Your-Access-Key outputUrl
https://licensing.accusoft.com/v1/WebDeployUser/WebDeployUser.aspx?16XTX23TT38QE85C6X33HR0tqh0yDCMPKq50aMej265M8SWM908
TIM969cEjEcgStH50ajaWg80
```

2. Copy the **URL** to a file, save it to a USB flash drive, and take it to a machine that has access to the Internet.
3. Copy the **URL** from the file on the USB flash drive and paste it into a browser. The following page is displayed:

[Home](#) > [Support](#) > [Licensing](#)

Accusoft Licensing Center

Welcome to Accusoft's deployment licensing site. This site allows you to manually register your product in cases where the target machine is not connected to the internet. Please see the product help file for additional information.

Browse our support area for developer resources to help build your applications better and faster.



Please enter the hardware key and click the button to generate the license.

Hardware Key:

16XTX23TT38QE85C6X33HR0tqh0yDCMPKq50aMej265M8SWM908

[Download license](#)

4. Click *Download license* button to download the license file.
5. Save the downloaded file on the USB flash drive, and take it back to the machine without internet access.
6. Copy the **License text** from the file on the USB flash drive and enter it in the command line and run the following command:

```
java.exe -jar plu.jar deploy write "Your Solution Name"
2.0.YourLicenseKeyTextFromUSBFlashDrive
```

7. The machine now has a deployment license installed.

Installing an Existing License Key Generated through the Web Portal

Usage

```
deploy write <solution name> <license key>
```

Parameters

- `<solution name>` – Solution name for deployment licensing. **Required.**
- `<license key>` – License key generated through the web portal. **Required.**

Examples

The following example demonstrates installing a deployment license generated through the web portal:

```
java.exe -jar plu.jar deploy write "Your Solution Name" 2.0.YourDeploymentLicenseKey
```

Configuring

This section covers common configuration options that you, as a PrizmDoc Viewer admin, will want to consider for PrizmDoc Server:

- [Central Configuration](#) - Information for configuring the cache, fidelity, file types, licensing, logging, networking, process ids, resource usage, security, user documents, viewing, work files, and PrizmDoc Cloud default values.
- [Implement Caching Strategies](#) - Discussion of cache management, optimizing cache performance, and cache strategy scenarios.
- [Adjust Caching Parameters](#) - Information to consider for setting cache lifetime, location, and reuse.
- [Change Encryption Keys for Public use Token Generation](#) - How to configure your own encryption keys (a security best practice).
- [Configure Microsoft Office Conversion](#) - How to configure a Linux deployment of PrizmDoc Server to delegate all Microsoft Office conversion work to a separate Windows deployment of PrizmDoc Server which has Microsoft Office installed.
- [Substitute Fonts for Office Rendering Fidelity](#) - Description of how PrizmDoc Server's logic selects the right font for rendering and recommendations for installing additional font packages.
- [Upgrade from Legacy Configuration](#) - How to migrate from legacy configuration to the new, single, central configuration file.

Central Configuration

Central Configuration

Configure PrizmDoc Server using the services configuration file for both Windows and Linux.

The configuration file will perform environment variable expansion in path values. The environment variable must be contained within a quoted string and enclosed with the % character. For example, "%ALLUSERSPROFILE%" or "%my_path%subpath" are both valid paths containing environment variables which will be expanded at runtime.

NOTES:

1. Given the default installation directory, the paths for the Central Configuration file are:
 - o **Linux:** /usr/share/prizm/prizm-services-config.yml
 - o **Windows:** C:\Prizm\prizm-services-config.yml
2. After making any changes to the configuration files, you need to **restart the PrizmDoc Server**. **IMPORTANT:** Any time you make a change to the configuration files that affects document output, you must clear your cache.
3. PrizmDoc Server uses central configuration by default (that is, watchdog.config contains the paths.central_config_file property, whose value is a path to the central configuration file relative to the PrizmDoc Server install directory).

The following options are available for configuration within the central configuration file:

NOTE: PrizmDoc Cloud uses specific default values that differ from the self-hosted default values as noted in the PrizmDoc Cloud Default Values section below.

- Cache
- Fidelity
- File Types
- License
- Logging
- Network
- Process Ids for API Process Resources
- Resource Usage
- Security
- User Documents
- Viewing
- Work Files
- JVM Options
- PrizmDoc Cloud Default Values

Cache

Property	Default Value	Supported Values	Description
cache.directory	<install_dir>/cache e.g.: /usr/share/prizm/cache C:\Prizm\cache	Any valid path to a directory with read and write permissions.	Directory where cache data is stored.

Fidelity

Property	Default Value	Supported Values	Description
<code>fidelity.svgMaxImageSize</code>	8000	Any integer greater than 0	For source documents which contain images, ensures that the images in the SVG delivered to the browser do not exceed a particular pixel width and/or height. For example, a value of 8000 would ensure that any images in a PDF whose width or height were greater than 8000 pixels would be down-sampled before the image was added to the final SVG. A typical value is 8000. The default value for this property is configurable. The out-of-box configuration uses a default value of 8000. Use 0 to disable the optimization.
<code>fidelity.vectorBackgroundColor.view.default</code>	white	CSS color name, e.g. "gray", or a hex RGB value, e.g. #FFFFFF	Defines the background color for viewing CAD documents if the background is not specified in the document. Also defines the background color for converting CAD documents to SVG, PNG, JPEG and TIFF. Please note, DGN documents define their background color, so this property does not affect them. Use <code>fidelity.vectorBackgroundColor.view.override</code> to define background color, ignoring the background color specified in the document.
<code>fidelity.vectorBackgroundColor.view.override</code>	none	CSS color name, e.g. "gray", or a hex RGB value, e.g. #FFFFFF, or "none"	Defines the background color for viewing CAD documents, ignoring the background specified in the document. Also defines the background color for converting CAD documents to SVG, PNG, JPEG and TIFF, ignoring the background specified in the document. Overrides <code>fidelity.vectorBackgroundColor.view.default</code> .
<code>fidelity.vectorTolerance</code>	0.3	A positive Floating point value with a minimum value of 0.0 and a maximum value of 10.0	For CAD documents, controls how much path simplification is allowed. The path simplification algorithm will merge points which are "close together" to create an optimized SVG. You can think of this value as defining what "close together" means. A typical value is 0.3. Higher values introduce more simplification, but also more distortion. The value cannot be greater than 10.0. The default value for this property is configurable. The out-of-box configuration uses a default value of 0.3. Use 0 to disable the optimization.
<code>fidelity.msOfficeDocumentsRenderer</code>	"auto"	One of the following strings: "auto", "libreoffice", and "msoffice"	Specifies the renderer to use with Microsoft Office documents. When set to "auto", PrizmDoc decides which renderer to use based on the MSO feature license state. If the MSO feature is present, then the Microsoft Office renderer will be used. Otherwise, Libreoffice will be used. When set to "libreoffice", PrizmDoc will be using the Libreoffice renderer. If you are intending to use the Libreoffice renderer permanently on a server that does not have Microsoft Office installed, contact info@accusoft.com to obtain the license key with the MSO feature disabled. When set to "msoffice", PrizmDoc will be using the Microsoft Office renderer, if the MSO feature is enabled in the license key, otherwise the licensing error will be reported. The parameter affects only Microsoft Office documents, RTF and CSV files. Other documents will continue to be rendered with LibreOffice. For a complete list of supported file types see Supported File Formats . NOTE: For PrizmDoc Cloud, the default value for <code>fidelity.msOfficeDocumentsRenderer</code> is "msoffice".
<code>fidelity.msOfficeCluster.host</code>	**	A valid IP address or hostname	This value is used to enable Microsoft Office Conversion connectivity for PrizmDoc servers running on Linux. Set this value on a PrizmDoc server running on Linux to the hostname (or the IP of a single PrizmDoc server, or a load balancer of a cluster running on Windows) to utilize the Microsoft Office Conversion service running on Windows to have native rendering of Microsoft documents in PrizmDoc.
<code>fidelity.msOfficeCluster.port</code>	0	Any open HTTP port on the server	This value is used to enable Microsoft Office Conversion connectivity for PrizmDoc servers running on Linux. Set this value on a PrizmDoc server running on Linux to the public port of a single PrizmDoc server (or a load balancer of a cluster running on Windows) to utilize the Microsoft Office Conversion service running on Windows to have native rendering of Microsoft documents in PrizmDoc. To connect to a single server specify <code>network.publicPort</code> parameter of the remote server. To connect to a load balancer specify <code>network.clustering.clusterPort</code> parameter of the cluster.

File Types

Property	Default Value	Supported Values	Description
<code>fileTypes.pdf.pageBoundaries</code>	mediaBox	One of the following strings: "mediaBox", "cropBox"	Controls which set of page boundaries should be used when interacting with PDF files. The PDF format specification defines five separate "Page Boundaries" that control various aspects of the imaging process. PrizmDoc Server supports "Media box" or "Crop box" to convert a source PDF document image.
<code>fileTypes.excel.margins.mode</code>	remove	One of the following strings: "preserve", "remove"	Controls how the page margins should be handled in Excel documents. "preserve" - Preserve (do not remove) document pages margins. Requires that <code>fileTypes.excel.pagination.dimensions.mode</code> be set to "preserve" and the Office documents renderer to be set to Microsoft Office (see <code>fidelity.msOfficeDocumentsRenderer</code> for more details). "remove" - Remove all margins from the document pages. Requires that <code>fileTypes.excel.pagination.dimensions.mode</code> be set to "override". NOTE: Changing this setting can affect rendered layout and page count. NOTE: For PrizmDoc Cloud, the default value for <code>fileTypes.excel.margins.mode</code> is "preserve".
<code>fileTypes.office.disableExternalHyperlinks</code>	false	true, false	Controls whether the external hyperlinks are enabled or disabled when viewing or converting Microsoft Office documents to PDF. false - All external hyperlinks are enabled. true - All external hyperlinks are disabled.
<code>fileTypes.excel.pagination.enabled</code>	true	true, false	Controls whether or not pagination is enabled for Excel documents. true (paginated mode) - Each Excel sheet is divided into a number of pages, such that each page fits into a specific size. false (non-paginated mode) - Each Excel sheet, irrespective of its size, is rendered onto a single page. If the number of columns and/or rows is large, then this might result in very small and unreadable output. NOTE: Changing this setting can affect rendered layout and page count.
<code>fileTypes.excel.pagination.dimensions.mode</code>	override	One of the following strings: "preserve", "override"	Controls which pagination mode to use if pagination is enabled for Excel documents (<code>fileTypes.excel.pagination.enabled</code> is set to "true"). "preserve" - Use the page dimensions specified in the Excel file. Requires that <code>fileTypes.excel.margins.mode</code> be set to "preserve" and the Office documents renderer to be set to Microsoft Office (see <code>fidelity.msOfficeDocumentsRenderer</code> for more details). "override" - Ignore the page dimensions of the Excel file and instead always use the following settings in this config file: <code>fileTypes.excel.pagination.dimensions.minWidth, fileTypes.excel.pagination.dimensions.maxWidth, fileTypes.excel.pagination.dimensions.minHeight, fileTypes.excel.pagination.dimensions.maxHeight</code> Requires that <code>fileTypes.excel.margins.mode</code> be set to "remove". NOTE: For PrizmDoc Cloud, the default value for <code>fileTypes.excel.pagination.dimensions.mode</code> is "preserve".
<code>fileTypes.excel.pagination.dimensions.minWidth</code>	"11.0in"	String consisting of a positive number followed by "in"	Controls the minimum page width for pagination of Excel files when <code>fileTypes.excel.pagination.dimensions.mode</code> is set to "override".
<code>fileTypes.excel.pagination.dimensions.maxWidth</code>	"22.0in"	String consisting of a positive number followed by "in"	Controls the maximum page width for pagination of Excel files when <code>fileTypes.excel.pagination.dimensions.mode</code> is set to "override".
<code>fileTypes.excel.pagination.dimensions.minHeight</code>	"8.5in"	String consisting of a positive number followed by "in"	Controls the minimum page height for pagination of Excel files when <code>fileTypes.excel.pagination.dimensions.mode</code> is set to "override".
<code>fileTypes.excel.pagination.dimensions.maxHeight</code>	"17.0in"	String consisting of a positive number followed by "in"	Controls the maximum page height for pagination of Excel files when <code>fileTypes.excel.pagination.dimensions.mode</code> is set to "override".
<code>fileTypes.excel.renderGridlines</code>	true	true, false	Specifies whether or not the Excel gridlines in all worksheets of the workbook should be rendered.
<code>fileTypes.excel.renderOnlyPrintArea</code>	false	true, false	Specifies whether the print areas defined in Excel workbook are to be honored or not. When set to "true", only the content defined within the print areas will be rendered. When set to "false", the content that goes beyond print areas will be rendered as well. NOTE: For PrizmDoc Cloud, the default value for <code>fileTypes.excel.renderOnlyPrintArea</code> is "true".

Property	Default Value	Supported Values	Description
<code>fileTypes.excel.renderHeadersAndFooters</code>	true	true, false	Specifies whether or not headers and footers of an Excel workbook should be rendered. When set to "true", even if the original document is missing the headers and footers, a space for headers and footers shall be reserved when rendering an Excel document.
<code>fileTypes.excel.renderHiddenContent</code>	true	true, false	Specifies whether or not the hidden rows, hidden columns, and whole spreadsheets that are hidden in the original Excel workbook are to be rendered.
<code>__experimental.fileTypes.email.renderMeetingInfo</code>	false	true, false	Controls whether email headers containing the meeting information (When, Who) should be rendered in email documents. NOTE: This feature is a work-in-progress that is not officially supported by Accusoft. Its behavior may change at any time in a future release of the product. We are collecting and reviewing any feedback you can provide about this feature at https://ideas.accusoft.com/ideas/PDV-1-745 NOTE: If you change the <code>__experimental.fileTypes.email.renderMeetingInfo</code> setting, you must clear your cache in order for the new settings to take effect on the files that were previously converted with the old rendering mode. NOTE: This setting affects the rendering layout of email documents. Markup generated for documents rendered with this setting turned off cannot not be used for documents rendered with this setting turned on and vice versa.

License

Property	Default Value	Supported Values	Description
<code>license.solutionName</code>	None (Required)	Valid solution name string	PrizmDoc Server solution name.
<code>license.key</code>	None (Required)	Valid license key string	PrizmDoc Server license key.

Logging

Property	Default Value	Supported Values	Description
<code>logging.directory</code>	<install_dir>/logs e.g.: /usr/share/prizm/logs C:\Prizm\logs	Any valid path to a directory with write permissions.	Directory where all log files will be stored. PrizmDoc Server is made up of several different processes, each of which create and maintain their own logs. The logs are invaluable for diagnosing issues with PrizmDoc Server if they arise. If you find yourself in this situation, please see the topic Packaging Log Files for Product Support to expedite your support request.
<code>logging.daysToKeep</code>	7	Any natural number	The number of rotated logs to keep in addition to the active log file. Logs are rotated each day at midnight (UTC). NOTE: This value does not currently apply to all services. Some services will always keep 7 rotated logs. Logs with 7 day archives: <ul style="list-style-type: none"> • AutoRedactionService.log • FormatDetectionService.log • HTMLConversionService.log • OfficeConversionService.log • PDFConversionService.log • RasterConversionService.log • VectorConversionService.log

Network

Property	Default Value	Supported Values	Description
<code>network.publicPort</code>	None (Required)	Any open HTTP port on the server	The public port the REST API will be available on. The chosen port must be accessible by all servers that need to call the PrizmDoc Viewer APIs.
<code>network.internalStartingPort</code>	None (Required)	Any open HTTP port on the server	The product requires a range of 200 ports which are reserved for its own internal use. This setting defines the starting port of that range (e.g. a value of 19000 means that ports 19000 through 19199 would be reserved for use by the product). These ports must not be accessible from outside of the server, for security reasons.
<code>network.clustering.enabled</code>	false	true, false	Set to true to enable cluster mode.
<code>network.clustering.clusterPort</code>	-	Any open HTTP port on the server	The port used to route requests to other servers in the cluster. This port needs to be exposed to the other servers in the cluster.
<code>network.clustering.servers</code>	-	Array containing hostnames or ip addresses of other servers within the cluster.	The server list can be set once via config, or repeatedly at runtime via a REST API call. Setting the list of servers here is useful if you have a static set of machines that will not change.

Process Ids

Property	Default Value	Supported Values	Description
<code>processIds.lifetime</code>	"20m"	Formatted Value, see Description	The length of time that a redaction creator, markup burner, content converter, form extractor, search context, search task or plain text redactor process remains usable. This must be an integer, followed by "s", "m", "h", or "d". The suffixes stand for second, minute, hour, or day, respectively. There should not be any space characters between the number and suffix. For example, "20m" indicates viewing sessions will timeout after 20 minutes. Please see the topic Implement Caching Strategies for more details. NOTE: For PrizmDoc Cloud, the default value for <code>processIds.lifetime</code> is 5h.

Resource Usage

Property	Default Value	Supported Values	Description
<code>resourceUsage.pccis.instances</code>	3	Any integer greater than 0	The number of PCCIS ASP.NET application instances to run concurrently.
<code>resourceUsage.ocs.numInstances</code>	"auto"	Any integer greater than 0, or the string "auto"	The number of LibreOffice conversion service instances to run concurrently, or "auto" to let the product choose an appropriate value. We recommend using the default value of "auto". If you do provide a specific value, it should not be set higher than the number of physical cores available on your server.
<code>resourceUsage.ocs.numThreads</code>	"auto"	Any integer greater than 0, or the string "auto"	The number of threads each instance should create to handle document processing requests, or "auto" to let the product choose an appropriate value. We recommend using the default value of "auto". If you do provide a specific value, it should not be higher than 2 x ocs.instances.
<code>resourceUsage.ocs.numPorts</code>	"auto"	Any integer greater than 0, or the string "auto"	The number of ports which can be used internally for communication with the LibreOffice conversion instances, or "auto" to let the product choose an appropriate value. We recommend using the default value of "auto". If you do provide a specific value, it should be no higher than 4 x ocs.instances.

Security

Property	Default Value	Supported Values	Description
security.aesEncryption.key	"E9rU73IZ2vd0he8Ls/hD8A=="	Base64 encoded value of a byte array representing an AES key with a size of 128, 192 or 256 bits.	The AES encryption key used to create external viewing session IDs. The external viewing session ID is a AES encrypted, Base64 encoded value of a string in the format of: <internal ID>/<server's host name>/<Auth-Token header value> Internal ID: This value is a unique GUID that is internally created by PCCIS for each new viewing session. Server's Host Name: Aptly named, this value is the hostname of the server on which PCCIS is running. Auth-Token Header Value: If the "Auth-Token" HTTP header exists in the initial POST request to create a viewing session, its value will be used here. Otherwise, "accusoft" is used. This value is useful if you have the need to store an authorization token for each viewing session which a proxy might need. See PrizmDoc Cluster Management for more details.
security.aesEncryption.iv	"JTN2XBjybtFA2fpsv6mylQ=="	Base64 encoded value of a byte array representing an AES initialization vector with a size of 128 bits.	The AES encryption initialization vector (iv) used to create external viewing session IDs.
security.htmlRendering.blockExternalContent	false	true, false	When rendering any source document which uses HTML content, controls whether or not externally-referenced content, such as images and iframes, will be blocked. This option affects any source document file type which uses HTML, including HTML, EML, and MSG. NOTE: Changing this setting can affect rendered layout and page count. NOTE: If you change the security.htmlRendering.blockExternalContent setting, you must clear your cache in order for the new settings to take effect on files previously converted with the old rendering mode. false - When rendering HTML, issue network requests for externally-referenced content in images, iframes, etc., and include that content in the final output. Any URL accessible from this machine may be loaded and included in the final output. See the Security Guidance page for more details. true - When rendering HTML, block externally-referenced content in images, iframes, etc. No network requests will be issued when rendering HTML and the final output will only include the content that is directly-present in the source HTML.

User Documents

Property	Default Value	Supported Values	Description
userDocuments.directory	<instal_dir>/cache/ UserDocuments e.g.: /usr/share/prizm/cache C:\Prizm\cache /UserDocuments	Any valid path to a directory with read permissions.	A directory that contains your documents for use when the documentSource viewing session property is set to "file".

Viewing

Property	Default Value	Supported Values	Description
viewing.allowDocumentDownload	false	true, false	Controls whether or not the REST API will accept requests to download the source document for a given viewing session.
viewing.sessionLifetime	"20m"	Formatted Value, see Description	The length of time that a viewing session remains usable. This must be an integer, followed by "s", "m", "h", or "d". The suffixes stand for second, minute, hour, or day, respectively. There should not be any space characters between the number and suffix. For example, "20m" indicates viewing sessions will timeout after 20 minutes. Please see the topic Implement Caching Strategies for more details. NOTE: For PrizmDoc Cloud, the default value for viewing.sessionLifetime is 5h.
viewing.cacheLifetime	"1d"	Formatted Value, see Description	The length of time that a document is cached and can be potentially reused by other new viewing sessions. This must be an integer, followed by "s", "m", "h", or "d". Those suffixes stand for second, minute, hour, or day, respectively. There should not be any space characters between the number and suffix. For example, "1d" indicates that data will be cached for up to one day. Please see the topic Implement Caching Strategies for more details. NOTE: If the amount of time specified by viewing.cacheLifetime is shorter than the amount of time specified by viewing.sessionConstraints.minSecondsAvailable.max, the viewing.cacheLifetime will be changed at runtime to match the amount of time specified by viewing.sessionConstraints.minSecondsAvailable.max. If you change this value to decrease the cache lifetime, make sure that you also adjust the viewing.sessionConstraints.minSecondsAvailable.max value to represent the same or shorter amount of time. NOTE: For PrizmDoc Cloud, the default value for viewing.cacheLifetime is 43h.
viewing.contentEncryption.enabled	false	true, false	Controls whether or not content is encrypted by the back end before being transmitted to a Viewer. The Viewer will decrypt the content in the browser. This is useful for DRM, making it more difficult to copy protected content that has been delivered to the browser.
viewing.sessionConstraints.documentSource.allowedValues	["api","http"]	Array which contains one or more of the following strings: "api", "http", "file"	Creates a value filter that will be applied to the "documentSource" JSON property when creating a new viewing session to ensure appropriate values are being set. If the actual property value fails to match the filter, an error will be returned and the viewing session will not be created. Allowing a combination of document sources here enables you to create viewing sessions with different sources on the fly without needed to modify this config file. This property is one of several that can be used to limit unwanted values from being used within the JSON properties of the initial POST request to create a viewing session. These filter values are useful to prevent mistaken or malicious values from being sent that could affect server behavior. NOTE: For PrizmDoc Cloud, the default value for viewing.sessionConstraints.documentSource.allowedValues is "api".
viewing.sessionConstraints.countOfInitialPages.min	0	Any natural number	Minimum allowed value for the "countOfInitialPages" JSON property when creating a new viewing session. Together with viewing.sessionConstraints.countOfInitialPages.max create a range filter that will be applied to ensure appropriate values are being set. If the actual property value fails to match the filter, an error will be returned and the viewing session will not be created.

Property	Default Value	Supported Values	Description
<code>viewing.sessionConstraints.countOfInitialPages.max</code>	10	Any natural number not less than <code>viewing.sessionConstraints.countOfInitialPages.min</code>	<p>This property is one of several that can be used to limit unwanted values from being used within the JSON properties of the initial POST request to create a viewing session. These filter values are useful to prevent mistaken or malicious values from being sent that could affect server behavior.</p> <p>Maximum allowed value for the "countOfInitialPages" JSON property when creating a new viewing session.</p> <p>Together with <code>viewing.sessionConstraints.countOfInitialPages.min</code> create a range filter that will be applied to ensure appropriate values are being set. If the actual property value fails to match the filter, an error will be returned and the viewing session will not be created.</p> <p>This property is one of several that can be used to limit unwanted values from being used within the JSON properties of the initial POST request to create a viewing session. These filter values are useful to prevent mistaken or malicious values from being sent that could affect server behavior.</p>
<code>viewing.sessionConstraints.documentExtension.regex</code>	".*"	Valid regular expression using the .NET Regular Expression Language	<p>Creates a regex filter that will be applied to the "documentExtension" JSON property when creating a new viewing session to ensure appropriate values are being set. If the actual property value fails to match the filter, an error will be returned and the viewing session will not be created.</p> <p>This property is one of several that can be used to limit unwanted values from being used within the JSON properties of the initial POST request to create a viewing session. These filter values are useful to prevent mistaken or malicious values from being sent that could affect server behavior.</p>
<code>viewing.sessionConstraints.externalId.regex</code>	".*"	Valid regular expression using the .NET Regular Expression Language	<p>Creates a regex filter that will be applied to the "externalId" JSON property when creating a new viewing session to ensure appropriate values are being set. If the actual property value fails to match the filter, an error will be returned and the viewing session will not be created.</p> <p>This property is one of several that can be used to limit unwanted values from being used within the JSON properties of the initial POST request to create a viewing session. These filter values are useful to prevent mistaken or malicious values from being sent that could affect server behavior.</p>
<code>viewing.sessionConstraints.pageContentEncryption.allowedValues</code>	["default"]	An array with either one or all of the following strings: "default", "enabled", "disabled"	<p>Creates a value filter that will be applied to the "pageContentEncryption" JSON property when creating a new viewing session to ensure appropriate values are being set. If the actual property value fails to match the filter, an error will be returned and the viewing session will not be created.</p> <p>This property is one of several that can be used to limit unwanted values from being used within the JSON properties of the initial POST request to create a viewing session. These filter values are useful to prevent mistaken or malicious values from being sent that could affect server behavior.</p>
<code>viewing.sessionConstraints.serverCaching.allowedValues</code>	["none","full"]	An array with one or more of the following strings: "none", "full"	<p>Creates a value filter that will be applied to the "serverCaching" JSON property when creating a new viewing session to ensure appropriate values are being set. If the actual property value fails to match the filter, an error will be returned and the viewing session will not be created.</p> <p>This property is one of several that can be used to limit unwanted values from being used within the JSON properties of the initial POST request to create a viewing session. These filter values are useful to prevent mistaken or malicious values from being sent that could affect server behavior.</p> <p>Please see the topic Implement Caching Strategies for more details.</p>
<code>viewing.sessionConstraints.render.alwaysUseRaster.allowedValues</code>	[false]	An array with one or more of the following values: true, false	<p>Creates a value filter that will be applied to the "render.html5.alwaysUseRaster" JSON property when creating a new viewing session to ensure appropriate values are being set. If the actual property value fails to match the filter, an error will be returned and the viewing session will not be created.</p> <p>This property is one of several that can be used to limit unwanted values from being used within the JSON properties of the initial POST request to create a viewing session. These filter values are useful to prevent mistaken or malicious values from being sent that could affect server behavior.</p>
<code>viewing.sessionConstraints.minSecondsAvailable.max</code>	86400	Any integer greater than 0	<p>This configuration property provides a maximum value that can be used for the option 'minSecondsAvailable'. The value is a positive number and represents seconds. When this value is set to zero, the ViewingSession timeout will be used for validation of the minSecondsAvailable option in the POST /viewingSession. The default will be 86400 seconds (1 day). This option is available for PrizmDoc version 12.0 or greater.</p>

Work Files

Property	Default Value	Supported Values	Description
<code>workFiles.directory</code>	cache.directory/WorkFileCache	Any valid path to a directory with read and write permissions.	Directory where work files are stored. This should be set to a non-shared location.
<code>workFiles.lifetime</code>	"1d"	Formatted Value, see Description	<p>The length of time that a workfile remains usable. This must be an integer, followed by "s", "m", "h", or "d". The suffixes stand for second, minute, hour, or day, respectively. There should not be any space characters between the number and suffix. For example, "20m" indicates viewing sessions will timeout after 20 minutes.</p> <p>Please see the topic Implement Caching Strategies for more details.</p>

JVM Options

Property	Default Value	Supported Values	Description
<code>eps.jvm.opts</code>	""	One or more options supported by JVM	<p>Java Virtual Machine settings to use when starting Email Processing Service.</p> <p>Examples:</p> <ul style="list-style-type: none"> "-Xms2G -Xmx5G" - Set initial JVM heap size to 2G and max heap size to 5G. "-XX:MaxRAMPercentage=50.0 -XX:+UseG1GC" - Set dynamic JVM heap size and use G1GC garbage collector.
<code>pdfps.jvm.opts</code>	""	One or more options supported by JVM	<p>Java Virtual Machine settings to use when starting PDF Processing Service.</p> <p>Examples:</p> <ul style="list-style-type: none"> "-Xms2G -Xmx5G" - Set initial JVM heap size to 2G and max heap size to 5G. "-XX:MaxRAMPercentage=50.0 -XX:+UseG1GC" - Set dynamic JVM heap size and use G1GC garbage collector.

PrizmDoc Cloud Default Values

PrizmDoc Cloud uses specific default values that differ from self-hosted default values as noted below:

- `fidelity.msOfficeDocumentsRenderer`: "msoffice"
- `processIds.lifetime`: 5h

- `fileTypes.excel.margins.mode:"preserve"`
- `fileTypes.excel.pagination.dimensions.mode:"preserve"`
- `fileTypes.excel.renderOnlyPrintArea:"true"`
- `viewing.cacheLifetime:43h`
- `viewing.sessionConstraints.documentSource.allowedValues:"api"`
- `viewing.sessionLifetime:5h`

PCCIS Configuration

PCCIS Configuration

This topic covers how to configure the PCCIS service of PrizmDoc Server. Most of the parameters are updated automatically as discussed in the [Central Configuration](#) topic. However, some parameters can be updated manually to configure viewing of large documents. The default timeout values might not be enough when viewing large documents, so the parameters described in this topic offer a way to increase those values.

NOTES:

1. Given the default installation directory, the paths for the Central Configuration file are:
 - **Linux:** `/usr/share/prizm/pccis/ServiceHost/pcc.config`
 - **Windows:** `C:\Prizm\PCCIS\ServiceHost\pcc.config`
2. After making any changes to the configuration files, you need to [restart the PrizmDoc Server](#).

The following options are available for configuration within the PCCIS configuration file:

Property	Default Value	Supported Values	Description
<code>PageInteractiveTimeout</code>	25000	An integer recommended to be between 5000 (5 seconds) and 120000 (2 minutes)	When PCCIS receives a request for page-level content or other data, this is the number of milliseconds that PCCIS will wait for that information to become available before the request times out and returns an error. Consider increasing this value if you are viewing documents containing a large number of pages.
<code>DocumentInteractiveTimeout</code>	50000	An integer recommended to be between 30000 (30 seconds) and 300000 (5 minutes)	When PCCIS receives a request for document-level data like page count, this is the number of milliseconds that PCCIS will wait for that information to become available before the request times out and returns an error. Consider increasing this value if you are viewing documents containing a large number of pages or comparing documents with a large number of differences.
<code>DocumentAcquisitionTimeout</code>	45000	An integer recommended to be between 5000 (5 seconds) and 120000 (2 minutes)	When PCCIS is responsible for downloading the source document directly from a specified HTTP location (<code>documentSource</code> viewing session property equals <code>"http"</code>), this is the number of milliseconds that request will wait before timing out. Consider increasing this value if you are viewing large (in terms of size) documents or comparing documents with a large number of differences.
<code>InternalOperationTimeout</code>	100000	An integer	When PCCIS begins the conversion for a

Property	Default Value	Supported Values	Description
		recommended to be between 100000 (100 seconds) and 600000 (10 minutes)	document, it has InternalOperationTimeout milliseconds to complete all of the conversion, comparison and text extraction operations. If this timeout is reached, the viewing session will be stopped because a valid document was not obtained. Consider increasing this value if you are comparing documents with a large number of differences or performing a text search having a large number of occurrences in the document.
RetainContextOnHealthIssue	false	true, false	A flag indicating whether the context (recently-processed work) should be saved when the service becomes unhealthy. Normally, this should be false, but you can set it to true to preserve documents that cause problems. When this is true, and only when the service transitions from healthy to unhealthy, the source documents and cached work will not be expired and deleted. This allows them to be reprocessed to see if they caused the health problem. IMPORTANT Note that if you stop and restart PCCIS, it will no longer preserve the files and may delete them.

Implement Caching Strategies

Introduction

This topic covers common questions and recommendations to consider when implementing your caching strategy:

- [Why does PrizmDoc Server Cache Files?](#)
- [What is the Cost of the PrizmDoc Server Cache?](#)
- [How do I Optimize Cache Performance?](#)
- [Cache Strategies and Tradeoff Scenarios](#)
- [Summary](#)

Why does PrizmDoc Server Cache Files?

The power behind PrizmDoc Services' ability to deliver viewable web content quickly and efficiently lies with its cache management. Viewing a multipage document requires that each document page be converted into a web compatible format such as JPEG, PNG or ideally SVG (which gives the highest fidelity upon scaling). Unfortunately, the conversion process is not instantaneous, which means there is some delay before a page can be made viewable. Because PrizmDoc Server assumes a document will be viewed by more than one person over multiple sessions, it converts all the pages into web viewable intermediate objects that are stored in its cache folders.

The conversion process begins when the viewing session is started or with the first request to view a document page by a given viewing session. Typically, the viewable page data that is generated will then be made available to any subsequent request for the same pages, reducing the time to view to only the time it takes to download the page data to the browser. To summarize, the cached files help deliver viewing performance because the viewing

objects are pre-generated and stored in the cache folders.

What is the Cost of the PrizmDoc Server Cache?

The cached files require storage on some media device for some period of time. Cached files created for viewing may take up a considerable amount of space, so there is a need to have some control on the growth of the cache files. Fortunately, PrizmDoc Server does provide ways to deal with the storage usage demand of the cache with options for controlling both where the files are stored, and how long they are stored there. In fact, the cache contains different purposed folders which can be relocated to different devices which can spread the cache burden out to different devices if necessary.

How do I Optimize Cache Performance?

The majority of the PrizmDoc Server cache is made up of pre-generated document pages which are readily available on demand. Caching these files is already a help in performance when the same document is viewed repeatedly. While there are three configurable cache folders locations, placing certain ones on more responsive media can result in better viewing experience with less burden on the server hosting the PrizmDoc Server service. The use of solid state drives (SSD) or Shared Memory (Linux only) minimizes input/output (I/O) latency and access times for cached files but these storage devices are typically much more confined in storage capacity.

Cache Strategies and Tradeoff Scenarios

Several scenarios are proposed below with purposed cache configuration solutions. The user should be familiar with the [central configuration](#) file settings as outlined in [Central Configuration Options](#). Along with the central configuration file, there is a property in the JSON object which the application posts when requesting a new viewing session from PrizmDoc Server (refer to the [How To Adjust Caching Parameters for PrizmDoc Server](#) topic).

The default settings in the central configuration file will cause viewing sessions to timeout after 20 minutes, and cached files to expire after one day. Also by default, the PrizmDoc Server cache folders will all be created within the same parent directory on the root drive. These default settings give a reader 20 minutes to read a document once the viewing session is started. After that time period, a new viewing session will need to be created for them to continue reading the document, either by refreshing their browser, or another mechanism you implement in your application.

The next time the same document is viewed, PrizmDoc Server will simply deliver the viewing objects that were created in the first viewing session to the same reader, or to any other reader viewing the same document, for about 24 hours after the first viewing session was created. When a reader (same or new) requests to read the document a day later, the cache process starts over because PrizmDoc will have already deleted the cached pages and will have to re-generate all the viewable content of the document again.

NOTE: *If you set the cache to 1 day, the timer will start over if someone accesses a file that is in the cache.*

To manually delete the cache:

1. Stop the **Prizm service**.
2. Go to the **cache folder**: (On Windows: C:\Prizm\cache. On Linux: /usr/share/prizm/cache).
3. You can delete all files and folders **within the cache folder**.
4. Start the **Prizm service** again. PrizmDoc will generate a new cache.

The file paths for the Central Configuration file are:

- **Linux:** /usr/share/prizm/prizm-services-config.yml
- **Windows:** C:\Prizm\prizm-services-config.yml

NOTE: The default installation directory is: C:\Prizm.

Scenario 1:

Viewing response appears slow even with caching enabled as lots of readers are interested in viewing the document.

Solution:

Set the `cache.directory` setting in the central configuration file to a faster SSD device or with Linux environments, set the content to a folder of the Shared Memory device (i.e. /dev/shm).

Example for Shared Memory Device

```
cache.directory: /dev/shm/Accusoft/Prizm/
```

The above setting in central configuration sets the cache directories to folders in Shared Memory on a Linux OS environment. Being faster than standard disk drives, PrizmDoc Server response will be typically quicker with less overall stress on the server to deliver viewing content.

Scenario 2:

Viewing Clients are getting errors and the storage device used for the PrizmDoc Server cache is showing errors because the devices are full.

Solution:

Depending on available storage capacity of the selected device, the cache expiration period specified by `viewing.cacheLifetime` in central configuration may need to be shortened to accommodate cache load. Please note that the time period for `viewing.cacheLifetime` should not be any shorter than the `viewing.sessionLifetime` time period. Otherwise, the `viewing.sessionLifetime` will take precedence and the cache expiration period will be forced to the same value. The `viewing.sessionLifetime` time period can be shortened but at the penalty of reducing the amount of time a user has to read a document in a single viewing session.

Rather than changing the viewing session timeout period, try changing the size of the (fast) storage device.

Example for Quicker Cache Cleanup

```
viewing.sessionLifetime: 15m  
viewing.cacheLifetime: 20m
```

The above settings set the viewing session timeout to 15 minutes and the life expectancy of any cached file to 20 minutes. After approximately 35 to 45 minutes, the cached files for a given document will be deleted. The exact time of cleanup can vary based on the scheduled nature of the cleanup processes and current load on the server.

Scenario 3:

Your application views a lot of large documents and users are not able to read them in time before they get a viewing session timeout error.

Solution:

The default setting in the central configuration file for `viewing.sessionLifetime` is 20 minutes. It can be increased to a larger value but that means PrizmDoc Server will have more resources to track at any given moment which could affect performance and host server capacity.

Example of Longer Viewing Session Duration

```
viewing.sessionLifetime: 1h
viewing.cacheLifetime: 1d
```

The above settings increase the ability for users to peruse a given document for an hour. Cache resources for the document will be removed 25+ hours later. As above, there is variability for cache cleanup based on the scheduled nature of the cleanup processes and current load on the server.

Scenario 4:

The documents served are fairly random and not typically shared with others.

Or:

The image is watermarked uniquely for each Viewer and should not be shared.

Solution:

In this scenario, the cache resources are not likely to be needed except for the initial user. There is a property in the JSON object which the application posts when requesting a new viewing session from PrizmDoc Server that can be used to disable caching on a per-viewing-session basis. The property, `serverCaching`, should be set explicitly to the string value `none` when the application requests a POST operation to get a new viewing session ID. Each document uploaded to PrizmDoc Server will be converted without PrizmDoc Server looking for an existing copy of the document. After the viewing session times out, the cached items for the document will be removed on a predetermined schedule which should be fairly quick because no other viewing sessions are using the data. For example:

Example

```
POST /ViewingSession
{
  ...
  "serverCaching": "none",
  ...
}
```

After the viewing session timeout, the cache items should be removed fairly soon.

Summary

The PrizmDoc Server cache provides a mechanism to deliver document content in a timely matter. However, each application is different and may tax server resources differently or have more demanding requirements. Balancing resource constraints against user experience can be a difficult task that may require compromises. Faster hardware, more specifically high speed storage devices, coupled with an understanding of the options for adjusting how the PrizmDoc Server cache behaves should allow you to reach a desired level of performance while maintaining a good user experience.

Adjust Caching Parameters

Introduction

When PrizmDoc Server receives a request to view a new document, also called a viewing session, it begins creating various artifacts associated with it and stores these artifacts on disk for a specified amount of time. These artifacts include such things as the original document, document metadata, and converted content used for viewing the document in a browser. We collectively refer to these artifacts as the PrizmDoc Server cache. This topic discusses various cache parameters that control things like cache lifetime, location, and reuse:

- [Cache Lifetime](#)
- [Cache Location](#)
- [Cache Reuse](#)

Cache Lifetime

The cache lifetime, or the amount of time cached files will exist on disk before being deleted for each new document, is controlled by two parameters, `viewing.sessionLifetime` and `viewing.cacheLifetime`. These parameters are found in the central configuration file.

The file paths for the Central Configuration file are:

- **Linux:** /usr/share/prizm/prizm-services-config.yml
- **Windows:** C:\Prizm\prizm-services-config.yml

NOTE: The default installation directory is: C:\Prizm.

Example

```
viewing.sessionLifetime: 20m
viewing.cacheLifetime: 1d
```

The session lifetime is the length of time that a viewing session remains usable. For example, this is the amount of time that a user can view and interact with a document in their browser before the document becomes unavailable. This value must be an integer followed by "s", "m", "h", or "d". The suffixes stand for second, minute, hour, or day, respectively. There should not be any space characters between the number and suffix. The example above indicates viewing sessions will timeout after 20 minutes.

The cache lifetime is the length of time that a document is cached and can be potentially reused by other new viewing sessions. This value must be an integer followed by "s", "m", "h", or "d". The suffixes stand for second, minute, hour, or day, respectively. There should not be any space characters between the number and suffix. The example above indicates that document data will be cached for one day.

IMPORTANT: If `viewing.sessionConstraints.minSecondsAvailable.max` is set to a value greater than `viewing.cacheLifetime` in the central configuration file the cache lifetime will be overwritten by the `minSecondsAvailable.max` value. This configuration property provides a maximum value that can be used for the option 'minSecondsAvailable' in the POST /ViewingSession. The value is a positive number and represents seconds. When this value is set to zero, the ViewingSession timeout will be used. The default is 86400 seconds (1 day).

The total cache lifetime of a document can be calculated by adding the session lifetime value to the maximum of either the session lifetime or cache lifetime.

This can be expressed as the following formula:

Total Cache Lifetime = Session Lifetime + max(Session Lifetime, Cache Lifetime)

The above formula will provide the approximate lifetime of a cached document because there is scheduling variability that can increase the actual time. This variability is caused partly by the periodic nature of which the cache cleanup processes are run. Also, if the server is under high load, the cleanup processes may be delayed so as not to consume additional resources.

NOTE: *If you set the cache to 1 day, the timer will start over if someone accesses a file that is in the cache.*

To manually delete the cache:

1. Stop the **Prizm service**.
2. Go to the **cache folder**: (On Windows: C:\Prizm\cache. On Linux: /usr/share/prizm/cache).
3. You can delete all files and folders **within the cache folder**.
4. Start the **Prizm service** again. PrizmDoc will generate a new cache.

See the [Caching Strategies](#) topic for details and recommendations for the viewing session timeout and cache expiration period values in your application.

Cache Location

The directories that PrizmDoc Server uses for caching are user-configurable. To change them, you will need to set the cache.directory parameter in the central configuration file. Cached files will be stored in subdirectories of this location.

Example

```
cache.directory: /usr/share/prizm/cache
```

See the [Caching Strategies](#) topic for details and recommendations on cache locations and types of storage media.

Cache Reuse

Consider the case where a viewing session is created in PrizmDoc Server, and PrizmDoc Server performs the work to convert the original source document to a format that is suitable for viewing in a browser. Now consider the case where multiple users are viewing the same document one or more times. PrizmDoc Server can leverage the cache in this case so that the document is converted only once but can be served for future viewing sessions using identical documents. The result is very fast viewing times for users and decreased processing time for servers.

PrizmDoc Server enables cache reuse by default. This property can be controlled per viewing session and is adjustable using the JSON properties of the initial POST request sent to PrizmDoc Server to create a viewing sessions.

Example

```
POST http://localhost:18681/Prizm_Services/V1/ViewingSession {
  "externalId": "MyDocumentName.pdf",
  "serverCaching": "Full",
  "render":
  {
    "html5":
```



```
    {
      "alwaysUseRaster":false
    }
  }
}
```

In the example above, the `serverCaching` property is set to a value of `Full`, which enables the reuse of the cache for multiple viewing sessions. This is the default value, so not including `serverCaching` at all would yield the same result. We recommend this feature to obtain the best performance.

To disable reuse of the cache, set `serverCaching` to a value of `None`. This means that any new viewing sessions will need to convert the source document into viewable content even if the same document has been converted during a previous viewing session. Also, because no other viewing session will be reusing the converted document, the document data associated with viewing session will typically be deleted immediately after the viewing session expires.

Change Encryption Keys for Public use Token Generation

Introduction

As part of the normal operation of the PrizmDoc Server API, ID values and tokens are created and provided to the user for use in the public API. Some of these values contain embedded information used for request routing which can include host names, IP addresses and ports of the servers hosting the PrizmDoc Server. This network information should only be relative to internally accessible servers. Nonetheless, the PrizmDoc Server will encrypt the information whenever it is embedded in public-use tokens using AES symmetric encryption and further encode the ciphertext to Base64 to create the new ID or token.

The PrizmDoc Server API ships configured with a default AES key and Initialization Vector (IV) so PrizmDoc Server will work "out-of-the-box". However, **it is recommended that you replace the default encryption values with those of your choosing** to maintain the highest level of security. The following steps describe how to fully replace the default AES keys with your own.

Step 1: Obtain an AES Key and Initialization Vector (IV)

1. First, you will need an **AES key** and **IV** that is unique to your organization. Following the AES standard, the key value can be 128, 192 or 256 bits and the IV value must be 128 bits.
2. Once you have the key and IV, they must both be **Base64** encoded so that they are in a format which can be easily stored in the configuration files of the PrizmDoc Server.
3. With a **Base64 encoded AES key** and **IV value** you can now begin updating the configuration files.

Step 2: Update the Central Configuration File

The file paths for the Central Configuration file are:

- **Linux:** /usr/share/prizm/prizm-services-config.yml
- **Windows:** C:\Prizm\prizm-services-config.yml

NOTE: The default installation directory is: C:\Prizm.

1. Open the **central config file**.
2. Set the **security.aesEncryption.key** and **security.aesEncryption.iv** properties to the Base64 encoded values you created in Step 1.

3. Save and exit the config file.
4. After making any changes to the configuration files, you need to [restart PrizmDoc Viewer](#).

Configure Microsoft Office Conversion Connectivity

Introduction

PrizmDoc Viewer provides Microsoft Office Conversion connectivity for PrizmDoc Servers running on Linux. While the Microsoft Office Conversion add-on requires PrizmDoc Server running on Windows, it is possible to configure PrizmDoc Servers running on Linux to utilize the Microsoft Office Conversion service to have native rendering of Microsoft documents in PrizmDoc Viewer.

Please see [Windows Requirements](#) and [Windows Installation](#) sections for information regarding installation of PrizmDoc Viewer with the Microsoft Office Conversion service.

The following steps describe how to enable Microsoft Office connectivity on the Linux server.

Single Server Mode

Linux Server Configuration

1. An **MSO enabled license** is required for the Microsoft Office Conversion connectivity for PrizmDoc Servers running on Linux.
2. Configure the following **two parameters** in the [prizm-services-config.yml](#):

fidelity.msOfficeCluster.host

Set this value on a PrizmDoc Server running on Linux to the hostname or the IP of a single PrizmDoc Server running on Windows. By setting this value, you can use the Microsoft Office Conversion service running on Windows to have native rendering of Microsoft documents in PrizmDoc Viewer.

fidelity.msOfficeCluster.port

Set this value on a PrizmDoc Server running on Linux to the public port of a single PrizmDoc Server running on Windows specified by the `network.publicPort` parameter of the remote server. By setting this value, you can use the Microsoft Office Conversion service running on Windows to have native rendering of Microsoft documents in PrizmDoc Viewer.

3. Restart **PrizmDoc Viewer**.

Note that if the `/etc/hosts` file of the Linux machine is either empty, or contains only localhost entry, such as:

```
127.0.0.1 localhost
```

the Linux server might not be able to connect to the Windows server. Make sure that the network of the Linux machine is configured properly. For instance, the `/etc/hosts` file should contain at least 2 entries:

```
127.0.0.1 localhost
127.0.0.1 <hostname-of-your-machine>
```

Cluster Mode

Linux Server Configuration

1. An **MSO enabled license** is required for the Microsoft Office Conversion connectivity for PrizmDoc Servers running on Linux.
2. Please configure following **two parameters** in the [prizm-services-config.yml](#):

fidelity.msOfficeCluster.host

Set this value on a PrizmDoc Server running on Linux to the hostname or the IP of a load balancer of a cluster running on Windows to utilize the Microsoft Office Conversion service running on Windows to have native rendering of Microsoft documents in PrizmDoc Viewer.

fidelity.msOfficeCluster.port

Set this value on a PrizmDoc Server running on Linux to the public port of a load balancer of a cluster running on Windows specified by 'network.clustering.clusterPort' parameter of the cluster to utilize the Microsoft Office Conversion service running on Windows to have native rendering of Microsoft documents in PrizmDoc Viewer.

3. Restart **PrizmDoc Viewer**.

Note that if the /etc/hosts file of the linux machine is either empty, or contains only localhost entry, such as:

```
127.0.0.1 localhost
```

the Linux server might not be able to connect to the Windows cluster. Please make sure, that the network of the linux machine is configured properly. For instance, the /etc/hosts file should contain at least 2 entries:

```
127.0.0.1 localhost
127.0.0.1 <hostname-of-your-machine>
```

File Formats

After performing the configuration steps above, a PrizmDoc Server running on Linux will convert all Microsoft Office documents through the Microsoft Office Conversion renderer from the PrizmDoc Server running on Windows.

Please refer to the [Supported File Formats](#) section for the information on the exact file formats supported by the Microsoft Office Conversion renderer.

Substitute Fonts for Office Rendering Fidelity

Introduction

PrizmDoc Server uses a sophisticated logic to select the right font for accurate text rendering within Microsoft Office documents in the LibreOffice-based rendering mode. The logic to locate and use the appropriate font(s) can be broken down into the following sequence:

1. PrizmDoc Server looks for pre-configured font substitutions, provided by PrizmDoc Viewer out-of-the-box. The configuration file defining the substitution in JSON format is located here:
2. **Windows:** `<install_directory>/conf/OfficeConversionService.Font.Windows.config`
3. **Linux:** `<install_directory>/conf/OfficeConversionService.Font.Linux.config`
4. PrizmDoc Server looks for the exact font match within the local PrizmDoc Viewer fonts directory containing the fonts installed by the PrizmDoc Viewer distribution.

5. PrizmDoc Server looks for the exact font match in the system fonts directory.
6. If none of the steps above solved the substitution, the PrizmDoc Server will dynamically look for the most appropriate font substitution within the PrizmDoc Viewer local and system fonts using the font metrics. This step might provide less accurate results due to the specifics of certain fonts. Therefore, when dealing with a wide range of fonts and languages, it is recommended to install additional font packages. Specifically on Linux systems, such packages as "msttcorefonts" can be helpful to improve substitution of MS core fonts.

For Asian language support on Linux systems, make sure to install corresponding language support in addition to the installed Asian fonts. Refer to the [Install Asian Fonts](#) topic for more information.

Upgrade from Legacy Configuration

Introduction

This information in this topic will help you upgrade to [Central Configuration](#) from the deprecated legacy configuration:

To use Central Configuration, watchdog.config must contain the paths.central_config_file property whose value is a path to the central configuration file relative to the PrizmDoc Server install directory. This is the default setting, but if you are migrating from legacy configuration, you should verify that this property is set correctly.

The following table maps the legacy file and property that corresponds to each central config property:

The file paths for the Central Configuration file are:

- **Linux:** /usr/share/prizm/prizm-services-config.yml
- **Windows:** C:\Prizm\prizm-services-config.yml

NOTE: The default installation directory is: C:\Prizm.

Central Configuration Property	Legacy Configuration File	Legacy Configuration Property	Special Transfer instructions
license.solutionName	watchdog.config	license.solutionName	-
license.key	watchdog.config	license.key	-
network.publicPort	watchdog.config	cep_port OR sep_port, see instructions	If running in cluster mode, the central config value corresponds to cep_port. If it is running in single-server mode, it corresponds to sep_port instead.
network.internalStartingPort	watchdog.config	internal_starting_port	-
network.clustering.enabled	watchdog.config	server_mode	A central config value of true corresponds to a server mode of "multi", and false corresponds to "single".
network.clustering.clusterPort	watchdog.config	sep_port, see instructions	The central config value maps to sep_port only if running in cluster mode.
network.clustering.servers	watchdog.config	cep_servers	Reduce the cep_servers array to an array of only the addresses of each server. The port of each will be assumed to be network.clustering.clusterPort.
security.aesEncryption.key	plb/pcc.config, redaction.config, workfile.config, contentconversion.config	ViewingSessionEncryptionKey OR encryptionKey OR affinityTokenKey, depending on file	-
security.aesEncryption.iv	plb/pcc.config, redaction.config, workfile.config, contentconversion.config	ViewingSessionEncryptionIv OR encryptionIv OR affinityTokenIv, depending on file	-
logging.directory	watchdog.config	paths.app_log_dir	Join the elements of paths.app_log_dir into a path string.
logging.daysToKeep	watchdog.config, plb/pcc.config	logging.streams.count, workfileService.logging.count, redactionService.logging.count, fileViewer.logging.count, contentConversionService.logging.count, errorReportingService.logging.count, OR logging.count, depending on file	-
cache.directory	servicehost/pcc.config	DocumentPath, GroupStateFolder, TempcachePath	-
workFiles.directory	watchdog.config	workfileService.workfileCache.path	-
workFiles.lifetime	watchdog.config	workfileService.workfileCache.expirationPeriod	-
userDocuments.directory	servicehost/pcc.config	UserDocumentFolder	-
processIds.lifetime	watchdog.config	redactionService.cache.expirationPeriod, contentConversionService.cache.expirationPeriod	-
viewing.allowDocumentDownload	servicehost/pcc.config	EnableSourceDocumentDownload	-
viewing.sessionLifetime	servicehost/pcc.config	ViewingSessionTimeout	-
viewing.cacheLifetime	servicehost/pcc.config	CacheExpirationPeriod	-
viewing.contentEncryption.enabled	servicehost/pcc.config	EncryptPageContent	-
viewing.sessionConstraints.documentSource.allowedValues	servicehost/pcc.config	ViewingSessionPropertyDocumentSource	Split ViewingSessionPropertyDocumentSource into an array of allowed sources.
viewing.sessionConstraints.countOfInitialPages.min	servicehost/pcc.config	ViewingSessionPropertyCountOfInitialPages	Set the central config value to the min value from ViewingSessionPropertyCountOfInitialPages.
viewing.sessionConstraints.countOfInitialPages.max	servicehost/pcc.config	ViewingSessionPropertyCountOfInitialPages	Set the central config value to the max value from ViewingSessionPropertyCountOfInitialPages.
viewing.sessionConstraints.documentExtension.regex	servicehost/pcc.config	ViewingSessionPropertyDocumentExtension	-

Central Configuration Property	Legacy Configuration File	Legacy Configuration Property	Special Transfer instructions
viewing.sessionConstraints.externalId.regex	servicehost/pcc.config	ViewingSessionPropertyExternalId	-
viewing.sessionConstraints.pageContentEncryption.allowedValues	servicehost/pcc.config	ViewingSessionPropertyPageContentEncryption	Set the central config value to an array of the values allowed by ViewingSessionPropertyPageContentEncryption.
viewing.sessionConstraints.serverCaching.allowedValues	servicehost/pcc.config	ViewingSessionPropertyServerCaching	Set the central config value to an array of the values allowed by ViewingSessionPropertyServerCaching.
viewing.sessionConstraints.render.alwaysUseRaster.allowedValues	servicehost/pcc.config	Html5RenderAcceptableRasterValue	Set the central config value to an array of the values allowed by Html5RenderAcceptableRasterValue.
fileTypes.pdf.pageBoundaries	PDFConversionService.<Platform>.config	useCropBox	Set the central config value to "cropBox" if useCropBox is true, or "mediaBox" if it is false.
fileTypes.excel.pagination.enabled	watchdog.config	officeConversionService.excelPagination	-
fileTypes.excel.pagination.dimensions.minWidth	watchdog.config	officeConversionService.excelPageWidthMin	Add 'in' to the end of the value in watchdog.config.
fileTypes.excel.pagination.dimensions.maxWidth	watchdog.config	officeConversionService.excelPageWidthMax	Add 'in' to the end of the value in watchdog.config.
fileTypes.excel.pagination.dimensions.minHeight	watchdog.config	officeConversionService.excelPageHeightMin	Add 'in' to the end of the value in watchdog.config.
fileTypes.excel.pagination.dimensions.maxHeight	watchdog.config	officeConversionService.excelPageHeightMax	Add 'in' to the end of the value in watchdog.config.
fileTypes.office.disableExternalHyperlinks	watchdog.config	officeConversionService.disableExternalHyperlinks	-
fidelity.msOfficeDocumentsRenderer	watchdog.config	officeConversionService.msOfficeDocumentsRenderer	-
resourceUsage.pccis.instances	watchdog.config	pccis_instances, see instructions	Set the central config value equal to the number of objects in the pccis_instances array.
resourceUsage.ocs.numInstances	watchdog.config	officeConversionService.officeInstanceCount -	-
resourceUsage.ocs.numThreads	watchdog.config	officeConversionService.threadCount	-
resourceUsage.ocs.numPorts	watchdog.config	officeConversionService.officePortCount	-

Clustering

Introduction

The PrizmDoc Services are designed to run out-of-the-box on a single server. In the single-server mode, the PrizmDoc Services are listening on port 18681 by default and fulfilling requests entirely on the same server. There is no additional configuration to run in single-server mode. This mode is recommended if you have only one server hosting the PrizmDoc Services.

If your application requires more bandwidth or processing power than one server can handle, the PrizmDoc Services provide a mode that enables request load balancing and routing across multiple servers hosting PrizmDoc Server. This topic discusses the requirements and considerations for running the PrizmDoc Services in cluster mode.

Additional topics that support cluster mode:

- [Optimizing Cache Performance for Cluster Mode](#)
- [Affinity Tokens & Cluster Mode](#)

Cluster Mode

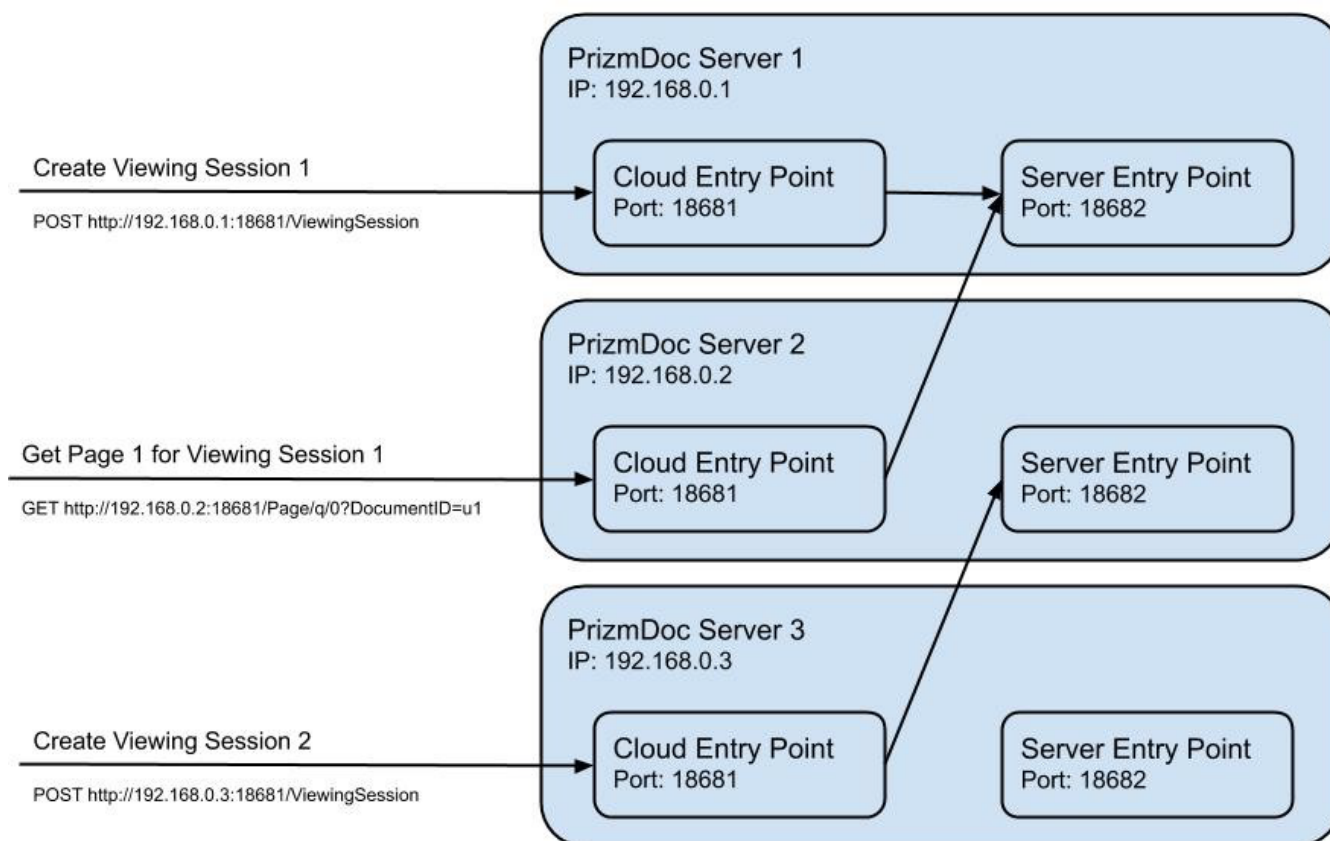
Before getting into the details of cluster mode, it's important to understand two things about how the PrizmDoc Server generates and serves content:

- The majority of requests the PrizmDoc Server handles are centered on document conversions or manipulations. These processes can be computationally expensive, so the PrizmDoc Server attempts to conserve as much CPU resources as possible by caching content as it's created.
- Cached content is only stored locally on the server where it was created and it is directly tied to a specific resource created by the RESTful web service. Requests for existing resources must be handled by the PrizmDoc Server on the server that originally created it, otherwise errors will occur.

How It Works

The cluster mode of the PrizmDoc Server works by creating a new entry point on each server hosting the PrizmDoc HTTP Service. This new entry point becomes responsible for routing requests to the correct PrizmDoc Server, as well as load balancing requests for new RESTful web service resources over all the PrizmDoc Servers in the node.

Consider the diagram below which depicts an architecture that employs 3 servers hosting the PrizmDoc HTTP Service within a node. Looking deeper, notice that each server is hosting two entry points:



The Server Entry Point (SEP) will be listening on port 18682 by default. This is the main PrizmDoc HTTP Service entry point for the server. It is responsible for routing requests to the internal services running on the server. It is also the same entry point that handles requests from your application in single-server mode. However, in cluster mode your application should not send requests directly to the SEP, but instead the requests should be made to the Cloud Entry Point.

The Cloud Entry Point (CEP) will be listening on port 18681 by default. In cluster mode, the CEP is responsible for routing requests to the correct PrizmDoc Server. If you are creating a new RESTful web service resource, the CEP will direct that request to a PrizmDoc Server it chooses.

If you are working with an existing resource, the CEP will ensure that the request is forwarded on to the server which originally created the resource. Any CEP can route any request to the correct server. This allows you to use a simple load balancer in front of your PrizmDoc Servers; simply have the load balancer send incoming requests to any CEP on any server and the CEP's will ensure that the requests are routed to the appropriate machine.

Step 1 - Configuration

After installation, the PrizmDoc HTTP Service will be running in single-server mode. To enable cluster mode:

1. Stop the **PrizmDoc Server**.
2. Open the **Central Configuration file** in a text editor.

The file paths for the Central Configuration file are:

- o **Linux:** /usr/share/prizm/prizm-services-config.yml
- o **Windows:** C:\Prizm\prizm-services-config.yml

NOTE: The default installation directory is: C:\Prizm.

3. Set the `network.clustering.enabled` value to `true`, and make sure the `network.publicPort` and `network.clustering.clusterPort` values exist and are assigned to valid port numbers:

```
network.clustering.enabled: true

network.clustering.clusterPort: 18682 # Server Entry Point for every server in
the cluster

network.publicPort: 18681 # Cloud Entry Point
```

4. Optionally, set the `network.clustering.servers` value to an array of address values corresponding to each PrizmDoc Server on the network node. For example:

```
network.clustering.servers: ["192.168.0.1", "192.168.0.2", "192.168.0.3"]
```

5. Save and close the **Central Configuration file**.
6. Start the **PrizmDoc Server**.

NOTE: If your application makes requests to the PrizmDoc service from another server, ports 18681 and 18682 (or other port values you choose) will need to be opened in the firewall for each server hosting the PrizmDoc service.

Step 2 - Start-Up

Once the PrizmDoc HTTP Service has been configured and is running on each server, there is one more critical step you must perform before the Cloud Entry Points will be able to handle requests successfully.

In this step you will inform the Cloud Entry Point on each PrizmDoc Server of the other available PrizmDoc Servers in the same network node. This list allows any CEP to route requests for existing resources to the correct PrizmDoc Server that originally created it, as well as load balance requests for new resources across all servers.

The list of servers is set by a HTTP PUT request to each Cloud Entry Point. Below is an example of the request that would be sent to each Cloud Entry Point (given the sample architecture shown in the diagram above):

Example

```
PUT http://192.168.0.1:18681/PCCIS/V1/Service/Properties/Servers {
  "servers": [
    {
      "address": "192.168.0.1",
      "port": "18682"
    },
    {
      "address": "192.168.0.2",
      "port": "18682"
    },
  ],
}
```

```
{
  {
    "address": "192.168.0.3",
    "port": "18682"
  }
}
```

This request would be repeated for the remaining two PrizmDoc Servers, the only change being the port specified in the HTTP request.

If the `network.clustering.servers` value was set in the **Central Configuration file** during the configuration step, the list of servers will automatically be initialized to this list on start-up. It may still be overridden by a HTTP PUT request to the Cloud Entry Point, but will initialize to the configured value again on subsequent start-ups unless the `network.clustering.servers` value is changed in the **Central Configuration file**.

Step 3 - Scaling

When PrizmDoc Servers are added or removed from the node, it is important that the list of servers held by each Cloud Entry Point is updated to reflect the servers that are actually active. Otherwise, requests will begin failing when routed to a server that does not exist, and new servers will not receive their fair share of new requests because not every PrizmDoc Server in the node is aware of them.

Keeping the server lists updated is a matter of repeating the requests described in the **Start-up** section above, only with updated JSON data that includes the new list of active servers.

Optimize Cache Performance for Cluster Mode

Introduction

Viewing Session resources require a number of conversions to create content that is suitable for viewing. Caching the converted content plays a large role in the performance of the PrizmDoc Server when handling requests for a Viewing Session. In single-server mode, caching is the most optimized because all converted content is available to the PrizmDoc Services on that server. As requests for new viewing sessions are received, the new documents are examined to locate existing content that may already be created if the same document was used in a previous viewing session.

In cluster mode, the PrizmDoc Services on each server maintain their own cache data, separate from other PrizmDoc Servers. Cache data is not shared across servers. Because of this, the effort to convert the same document will likely be duplicated on multiple PrizmDoc Servers if you have a situation where users are frequently viewing the same document more than once.

To counteract this side-effect of cluster mode, the PrizmDoc Services provide a way to increase the chances that a request for a new viewing session will be sent to the same PrizmDoc Server that may have already converted the same document. This is done by providing a "hint" value in a HTTP header of the request to create new viewing sessions. Below is an example request that sets this header:

Example

```
POST http://192.168.0.1:18681/PCCIS/V1/ViewingSession Accusoft-Affinity-Hint: my-unique-document-name.docx
{
  "tenantId": "my application name",
  "externalId": "my-unique-document-name.docx",
  "render": {
    "html5": {
```



```
        "alwaysUseRaster": false
    }
}
}
```

The Accusoft-Affinity-Hint header should only be specified in the HTTP request that creates a new viewing session.

It is important to note that the value used in this header should uniquely identify the document. A document name or a unique ID from a database are good options. Using the same value for documents that are not identical will unbalance the requests across the PrizmDoc Servers as it will cause a single server to be favored for all requests that contain the same hint value.

Note that the Accusoft-Affinity-Hint header is unnecessary when creating a viewing session through the PAS API. Refer to the [PAS Services Cluster Environments](#) topic for more information on cache optimization for PAS.

Affinity Tokens & Cluster Mode

Introduction

This topic covers requests for work files, markup burners, redaction creators, and content converters that require affinity tokens.

In cluster mode only, requests for WorkFile, MarkupBurner, RedactionCreator, and ContentConverter resources require an additional bit of data, called an affinity token, to ensure they are routed correctly by the Cloud Entry Point (CEP). The affinity token is a Base64 encoded string that contains encrypted information necessary to route related requests to the same PrizmDoc Server. Getting related requests to the same server is critical as the necessary data is cached locally.

In cluster mode, the PrizmDoc Server API will automatically generate an affinity token when it receives a POST request for a new ViewingSession, WorkFile, MarkupBurner, RedactionCreator or ContentConverter resource and return it in the response. Once you have obtained an affinity token, you will need to pass this in with related requests using the "Accusoft-Affinity-Token" HTTP custom header. In the case of a ViewingSession, the affinity token is only required when retrieving or setting the source document WorkFile ID for that session.

The following example request and response sequence demonstrates how an affinity token is used to burn markup into a document. Notice how all requests, (except the first request), use the same affinity token even for subsequent POST requests.

Example

```
// 1. Create first work file, the source PDF document to burn-in markup. POST
http://192.168.0.1:18681/PCCIS/V1/WorkFile?FileExtension=pdf Content-Type:
application/octet-stream
[binary data]
200 OK
Content-Type: application/json
{
  "fileId": "5qTYa3gzN9gYUb5SzqUhgq",
  "affinityToken": "rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w="
}
// 2. Create the second work file, the XML markup to burn-in.
// Pass in the affinity token generated from the previous request so that this
// XML data is stored on the same PCC server. POST
http://192.168.0.1:18681/PCCIS/V1/WorkFile?FileExtension=xml Accusoft-Affinity-
Token: rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w=
Content-Type: application/xml
```

```
[xml data]
200 OK
Content-Type: application/json
{
  "fileId": " 01bLJwFGxf9QGUTkyrOqig",
  "affinityToken": "rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w="
}
// 3. Create the markup burner resource, again on the same PCC server where we
created
//      the work files in the previous requests. POST
http://192.168.0.1:18681/PCCIS/V1/MarkupBurner Content-Type: application/json
Accusoft-Affinity-Token: rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w=
{
  "input": {
    "documentFileId": "5qTYa3gzN9gYUb5SzqUhqg",
    "markupFileId": "01bLJwFGxf9QGUTkyrOqig"
  }
}
200 OK
Content-Type: application/json
{
  "processId": "Rr64ma-U_HseoPrs6y0iiw",
  "expirationDateTime": "2014-12-03T18:30:49.460Z",
  "input": {
    "documentFileId": "5qTYa3gzN9gYUb5SzqUhqg",
    "markupFileId": " 01bLJwFGxf9QGUTkyrOqig"
  },
  "state": "processing",
  "percentComplete": 0,
  "affinityToken": "rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w="
}
// 4. Get status of markup burning process until state is "complete". GET
http://192.168.0.1:18681/PCCIS/V1/MarkupBurner/Rr64ma-U_HseoPrs6y0iiw Accusoft-
Affinity-Token: rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w=
200 OK
Content-Type: application/json
{
  "processId": "Rr64ma-U_HseoPrs6y0iiw",
  "expirationDateTime": "2014-12-03T18:30:49.460Z",
  "input": {
    "documentFileId": "5qTYa3gzN9gYUb5SzqUhqg",
    "markupFileId": " 01bLJwFGxf9QGUTkyrOqig"
  },
  "state": "complete",
  "percentComplete": 100,
  "output": {
    "documentFileId": "5ufb3ytUb1BxxgSUAk_G9Q"
  },
  "affinityToken": "rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w="
}
// 5. Get the burned document once the markup burning process is complete GET
http://192.168.0.1:18681/PCCIS/V1/WorkFile/5ufb3ytUb1BxxgSUAk_G9Q Accusoft-
Affinity-Token: rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w=
200 OK
Content-Type: application/pdf
[binary data]
```

Likewise, the following request and response sequence demonstrates how an affinity token is used in a content conversion workflow:

Example

```
// 1. Create first work file, the source PDF document to be converted. POST
http://192.168.0.1:18681/PCCIS/V1/WorkFile?FileExtension=docx Content-Type:
application/octet-stream
[binary data]
200 OK
Content-Type: application/json
{
  "fileId": "5qTYa3gzN9gYUb5SzqUhqg",
  "affinityToken": "rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w="
}
// 2. Create the content converter process. Pass in the affinity token generated
// from the previous request so that resource is stored on the same PCC
server. POST http://192.168.0.1:18681/v2/contentConverters Accusoft-Affinity-
Token: rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w=
{
  "input": {
    "src": {
      "fileId": 5qTYa3gzN9gYUb5SzqUhqg
    },
    "dest": {
      "format": "pdf"
    }
  }
}
200 OK
Content-Type: application/json
{
  "input": {
    "src": {
      "fileId": 5qTYa3gzN9gYUb5SzqUhqg
    },
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    }
  },
  "expirationDateTime": "2015-12-17T20:38:39.796Z",
  "processId": "E1kNzWtrUJp4rXI5YnLUgw",
  "state": "processing",
  "percentComplete": 0,
  "affinityToken": "rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w="
}
// 3. Get status of content conversion process until state is "complete". GET
http://192.168.0.1:18681/v2/contentConverters/E1kNzWtrUJp4rXI5YnLUgw Accusoft-
Affinity-Token: rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w=
200 OK
Content-Type: application/json
{
  "input": {
    "src": {
      "fileId": 5qTYa3gzN9gYUb5SzqUhqg
    },
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    }
  }
}
```

```
    }
  },
  "expirationDateTime": "2015-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "state": "complete",
  "percentComplete": 100,
  "output": {
    "results": [
      {
        "fileId": "KOrSwaqsguevJ97BdmUbXi",
        "src": [{ "fileId": "5qTYa3gzN9gYUb5SzqUhqg", "pages": "1-3" }]
      }
    ]
  }
}
"affinityToken": "rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w="
}
// 4. Get the converted document once the content conversion process is complete
GET http://192.168.0.1:18681/PCCIS/V1/WorkFile/KOrSwaqsguevJ97BdmUbXi Accusoft-
Affinity-Token: rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w=
200 OK
Content-Type: application/pdf
[binary data]
```

Start & Stop PrizmDoc Server

This section contains the following information:

- [Linux](#)
- [Windows](#)

Linux

Introduction

The included script **./scripts/pccis.sh** can be used to start and stop the Watchdog service, which will bring up the PrizmDoc Server as defined in the configuration files. The following examples assume the default install location. If you did not install it to the default location, replace **/usr/share/prizm** with the location to which you installed it.

Start

Example

```
/usr/share/prizm/scripts/pccis.sh start
```

Stop

Example

```
/usr/share/prizm/scripts/pccis.sh stop
```

Adding PCCIS to the Boot Sequence

You can also configure PCCIS to be started / stopped together with the system in 2 steps:

1. Create a symbolic link to the `./scripts/pccis.sh` in `/etc/init.d/` directory:

Example

```
ln -s /usr/share/prizm/scripts/pccis.sh /etc/init.d/pccis
```

2. Register PCCIS as an init script, so that it is managed by the system. This step is platform-dependent.

RedHat / CentOS

Example

```
chkconfig --add pccis
```

Ubuntu

Example

```
update-rc.d pccis defaults
```

Once done, the system should stop PCCIS, when going to reboot (runlevel 6) or shutdown (runlevel 0). It will also not be started, when booting in single-user mode (runlevel 1 - usually used for system recovery). PCCIS will be started in all other cases (runlevels 2 - 5).

TIP: Normally, there shouldn't ever be a need to change these defaults, but in case there is, you can use the mentioned commands to adjust the order of executing relative to other init scripts as well as runlevel manually:

Example

```
update-rc.d pccis start|stop NN runlvl [runlvl] [...]  
chkconfig [--level <levels>] <name> <on|off|reset|resetpriorities>
```

TIP: After the first step, you can use the **service** command to manage PCCIS instead of invoking the script directly. That way even if you don't need PCCIS to be automatically started / stopped, you may want to complete the first step to be able to more easily manage it. The syntax is as follows:

Example

```
service pccis start|stop|restart|status
```

Excluding PCCIS from the Boot Sequence

If you want to prevent PCCIS from starting / stopping together with the system, you need to revert Step 2 from the section above. It is done as follows:

RedHat / CentOS

Example

```
chkconfig --del pccis
```

Ubuntu

Example

```
update-rc.d -f pccis remove
```

Windows

Method 1: From the Windows Service

On Windows, the PrizmDoc Server should ideally be started/stopped from the Windows service management console. As part of the PrizmDoc Server installation, the service should be configured to start automatically. If you need to start, stop, or restart, use the following instructions:

1. Log onto the system using an account with administrator privileges.
2. Click **Start > Run**.
3. Type **services.msc**.
4. Press **Enter**.
5. Find **Prizm** in the list of services, and right-click on the service to access the context menu.
6. To Start the Service: Click **Start** and wait for the service to start. The status should update to "started" (this option will only be available if the service is not running).
7. To Stop the Service: Click **Stop** in the right-click menu and wait for the service control dialog. The status will be updated to be blank (this option will only be available if the service is already started).
8. To Restart the Service: Click **Restart** and wait for the service control dialog (this option will only be available if the service is already started).
9. Click **Start > Run**.
10. Type **inetmgr**.
11. Press **Enter**.
12. Find **PrizmDoc Web Site** under Sites in the tree view to the left.
13. Select that node and find options to start/stop/restart in the pane to the right.

Method 2: From the Command Line

If access to the control panel is not available, services can also be started/stopped from the command line using the following commands:

Example

```
net start Prizm
net stop Prizm
```

Service Logs

The Prizm Windows service will log certain status messages to the Windows Event Log. These messages can be helpful in diagnosing problems while starting and stopping the service. To view the Windows Event Log, use the following instructions:

1. Log onto the system using an account with administrator privileges.
2. Click **Start > Run**.
3. Type **eventvwr**.
4. Press **Enter**.
5. Expand Applications and Services Logs.
6. Click **Prizm**.

PrizmDoc Application Services

This section contains the following information:

- [Server Sizing](#)
- [Installing](#)
 - [Using Docker](#)
 - [Install on Windows](#)
 - [Windows Requirements & Supported Environments](#)
 - [Unattended Install & Uninstall](#)
 - [Installing with Traditional Linux Install Packages \(deprecated\)](#)
 - [Requirements & Supported Environments for Traditional Linux Install Packages](#)
 - [Uninstall Traditional Linux Install Packages](#)
 - [Check the Connection to PrizmDoc Server](#)
- [Configuring](#)
 - [PAS Configuration](#)
 - [PAS Database Administration & Maintenance](#)
 - [How & When to use CORS](#)
- [Clustering](#)
 - [Optimize Cache Performance for Cluster Environments](#)
 - [Run PAS on Clusters](#)
- [Starting & Stopping](#)
 - [Linux](#)
 - [Windows](#)

Server Sizing

Introduction

The [PrizmDoc Application Service](#) (PAS) can run on a wide variety of server configurations. It will automatically scale to utilize available cores, and payload data is streamed as much as possible to reduce RAM usage. It is typically external dependencies like the PrizmDoc Server, which PAS uses for process-heavy conversions that can affect its perceived responsiveness.

PAS may also be [configured to run on multiple servers](#). This, combined with modest hardware requirements, means that smaller, low-cost servers can be used and scaled horizontally (more servers are added) when additional capacity is needed.

Network Throughput

Many requests handled by PAS will result in a HTTP request to PrizmDoc Server. If PAS and PrizmDoc Server are hosted on separate servers, it is important to have good network throughput between nodes for optimal performance.

Example Server Configurations

The following minimum requirements can reasonably support 500 native PAS requests per second. A native PAS request is one that PAS handles entirely and does not depend on a supporting resource like PrizmDoc Server to fulfill the request.

Minimum Requirements

- Logical Cores: 1
- Memory: 0.5 GB (Linux) / 1 GB (Windows)
- Drive Type: SSD
- AWS: t2.nano (Linux) / t2.micro (Windows)

Suggested Requirements

- Logical Cores: 2
- Memory: 4 GB (Linux) / 8 GB (Windows)
- Drive Type: SSD
- AWS: t2.medium (Linux) / t2.large (Windows)

Additional Considerations for Pre-Conversion Services

When enabling the [Viewing Packages](#) feature, there are additional considerations for hardware requirements. This feature requires a database and will cause PAS to make additional HTTP requests to PrizmDoc Server, increase Disk IO, and execute queries against the database.

Network Throughput

If the database is hosted on a separate server, it is important to have good network throughput between nodes for optimal performance.

File Storage

PAS will use the file system to store all artifacts of a viewing package which increases the amount of storage needed by the service. The amount of storage needed is very closely tied to the number of active viewing packages and the types of documents being viewed. For example, a viewing package of a 100 page document requires several times the storage that a 1 page document needs. Likewise, a document page that contains drawings and

images will consume more storage than a page with just text. Testing with a representative set of sample documents is recommended to understand the file storage requirements of a specific application.

Database

PAS will store various bits of metadata about each viewing package in the configured database. The total amount of data stored for a viewing package in the database partly depends on the number of pages in the source document. However, the metadata size is not currently affected by the content of individual pages. Again, testing with a representative set of sample documents is recommended.

Example Server Configurations

The following minimum requirements can reasonably support 5 simultaneous viewing package creation processes per logical core. However, viewing package creation is heavily dependent on PrizmDoc Server for generating content. As PAS instances are scaled up, the supporting PrizmDoc Server instances must be scaled up appropriately to avoid overloading them. Additionally, the overall number of simultaneous viewing package creation processes should be throttled to avoid overburdening the system.

Minimum Requirements

- Logical Cores: 2
- Memory: 4 GB (Linux) / 8 GB (Windows)
- Drive Type: SSD
- AWS: t2.medium (Linux) / t2.large (Windows)

Suggested Requirements

- Logical Cores: 2
- Memory: 8 GB
- Drive Type: SSD
- AWS: m4.large

Installing

This section covers how to deploy an instance of PAS:

- [Using Docker](#)
- [Installing on Windows](#)
 - [Windows Requirements & Supported Environments](#)
 - [Unattended Install & Uninstall](#)
- [Installing with Traditional Linux Install Packages \(deprecated\)](#)
 - [Requirements & Supported Environments for Traditional Linux Install Packages](#)
 - [Uninstall Traditional Linux Install Packages](#)
- [Checking the Connection to PrizmDoc Server](#)

Using Docker

Using Docker

This section explains how to deploy a PAS instance using the official PAS docker image, available on Docker Hub as [accusoft/prizmdoc-application-services](#).

Note that PAS requires a connection to PrizmDoc Server. For the rest of this section, we assume that you already have a PrizmDoc Server instance or tier deployed. If that's not the case, you might be interested in our [Minimal Backend Quick Start](#) which explains how you can quickly deploy a single instance of PAS and PrizmDoc Server together.

Requirements

To run PAS as a Docker container, you simply need a Docker host (a machine with Docker installed). See the [Docker documentation](#) for more information.

1. Create a PAS config file

Before you can run PAS, you'll need a config file. We've included a special `init-config` command in our Docker image which you can use to create an initial config file.

Windows (PowerShell)

First, make sure you've created a `config` directory on your host file system. This will be the directory where your new config file will be created:

```
mkdir config
```

Then, use the Docker image's `init-config` command to create a new PAS config file:

```
docker run --rm -e ACCEPT_EULA=YES --volume $pwd/config:/config  
accusoft/prizmdoc-application-services init-config
```

This will create a new PAS config file on your Windows host filesystem at `.\config\pcc.nix.yml`.

Linux (bash)

Use the Docker image's `init-config` command to create a new PAS config file:

```
docker run --rm -e ACCEPT_EULA=YES --volume $(pwd)/config:/config  
accusoft/prizmdoc-application-services init-config
```

This will create a new PAS config file on your host filesystem at `./config/pcc.nix.yml`.

2. Customize your PAS config file

You will need to customize the contents of the `pcc.nix.yml` file for your PAS deployment.

NOTE: On a Linux system, because the config file was created by a Docker container, you will need to either edit the config file as root or change the owner of the file before editing it.

Typically, you need to:

- Change the `secretKey` to your own value (see [PAS Configuration](#) for more information).

- Configure your connection to PrizmDoc Server by updating the `pccServer` properties:

```
pccServer.hostName: your-prizmdoc-server-host
pccServer.port: 18681
pccServer.scheme: http
```

- Configure your storage options.

For more information, see [PAS Configuration](#).

3. Start PAS

Assuming you have configured PAS to run on port `3000` and use `./config`, `./logs`, and `./data` directories, you can start PAS like so:

Windows (PowerShell)

First, create directories for logs and data:

```
mkdir logs
mkdir data
```

Then, start a `pas` container:

```
docker run --rm --env ACCEPT_EULA=YES --publish 3000:3000 --volume
$(pwd)/config:/config --volume $(pwd)/logs:/logs --volume $(pwd)/data:/data --name pas
accusoft/prizmdoc-application-services
```

Linux (bash)

Start a `pas` container like so:

```
docker run --rm --env ACCEPT_EULA=YES --publish 3000:3000 --volume
$(pwd)/config:/config --volume $(pwd)/logs:/logs --volume $(pwd)/data:/data --
name pas accusoft/prizmdoc-application-services
```

In the examples above:

- `--rm` ensures the container is automatically deleted when it stops.
- `--env ACCEPT_EULA=YES` indicates you have accepted the [PrizmDoc Viewer license agreement](#).
- `--publish 3000:3000` publishes the container's port to the host. This assumes PAS was configured to use port `3000`. If you have configured PAS to use a different port, adjust accordingly.
- `--volume $(pwd)/config:/config` maps a host config directory into the container. Your local config directory must contain the `pcc.nix.yml` config file created earlier.
- `--volume $(pwd)/logs:/logs` maps a local logs directory into the container. After the container stops, the logs will remain in this directory.
- `--volume $(pwd)/data:/data` maps a local data directory into the container. After the container stops,

the data will remain in this directory and can be used again when restarting the container.

- `--name pas` sets the name of the running container.
- `accusoft/prizmdoc-application-services` is the image that should be run.

If you want to start the Docker container in the background, add the `-d` option to `docker run` to run the container in disconnected mode.

4. Check PAS health

After starting PAS, `GET http://localhost:3000/health` should return HTTP 200, indicating PAS is healthy. If you visit this URL in a browser, you should see "OK" in the body of the page.

Windows (PowerShell)

```
Invoke-WebRequest -Uri http://localhost:3000/health
```

Should output something like:

```
StatusCode      : 200
StatusDescription : OK
Content         : {79, 75}
RawContent      : HTTP/1.1 200 OK
                  Connection: keep-alive
                  Content-Length: 2
                  Date: Thu, 21 Nov 2019 17:49:35 GMT

                  OK
Headers         : {[Connection, keep-alive], [Content-Length, 2], [Date, Thu,
21 Nov 2019 17:49:35
                  GMT]}
RawContentLength : 2
```

Linux (bash)

```
curl -i http://localhost:3000/health
```

Should output something like:

```
HTTP/1.1 200 OK
Date: Wed, 20 Nov 2019 20:00:39 GMT
Connection: keep-alive
Content-Length: 2

OK
```

5. Check PAS's connection to PrizmDoc Server

After starting PAS, GET `http://localhost:3000/servicesConnection` should return HTTP 200, indicating PAS is configured correctly to make requests to PrizmDoc Server. If you visit this URL in a browser, you should see "OK" in the body of the page.

Windows (PowerShell)

```
Invoke-WebRequest -Uri http://localhost:3000/servicesConnection
```

Should output something like:

```
StatusCode      : 200
StatusDescription : OK
Content         : {79, 75}
RawContent      : HTTP/1.1 200 OK
                  Connection: keep-alive
                  Content-Length: 2
                  Date: Thu, 21 Nov 2019 17:50:11 GMT

                  OK
Headers         : {[Connection, keep-alive], [Content-Length, 2], [Date, Thu,
21 Nov 2019 17:50:11 GMT]}
RawContentLength : 2
```

Linux (bash)

```
curl -i http://localhost:3000/servicesConnection
```

Should output something like:

```
HTTP/1.1 200 OK
Date: Wed, 20 Nov 2019 20:00:39 GMT
Connection: keep-alive
Content-Length: 2

OK
```

5. Stopping the container

You can stop your named container with:

```
docker stop pas
```

Install on Windows

Installing PAS on Windows

PAS is delivered as part of the "PrizmDoc Client" installer. If you have questions about requirements before installing, refer to the [System Requirements & Supported Environments](#) topic.

NOTE: *If you have an updated license, you must re-start PAS and PrizmDoc Server in order to use the new license.*

To install PAS, follow these steps:

1. Download the "**PrizmDoc Client**" installer from the [website](#).
2. Double-click on the **PrizmDocClient-xx** application file to launch the installer (where xx represents the version). Click **Install PrizmDoc Client**.
3. Carefully read the information contained in the License Agreement form before making a decision to accept the terms of the agreement. Choose **I accept the terms in the License Agreement** to accept the conditions outlined in the License Agreement and then click **Next** to continue the installation (or click **Cancel** to exit the installation process).
4. The Select Features dialog is displayed. The Select Features dialog allows you to define what components of PrizmDoc Viewer you want to install:
 - **Legacy Samples** – Installs various legacy samples.
 - **PAS (PrizmDoc Application Services)** - Installs PAS.
 - **Configure ASP.NET Samples with IIS** - This option is only available if both IIS and ASP.NET 4.0+ are present.
 - **Re-register ASP.net 4.0 with IIS** - This option is only available if both IIS and ASP.NET 4.0+ are present.

Once you have made your selections, click **Next** to continue.

5. If you are installing the PAS (PrizmDoc Application Services), you have the option to enter the PrizmDoc Viewer **server address** to test the connection. The default PrizmDoc Server address is shown in the text field. You may also skip this step, and configure the server address manually later on.

Enter the **server address** for your PrizmDoc Server into the text field provided. Click **Test** to verify access to your PrizmDoc Server.

A valid URL for the PrizmDoc Server will include http or https, a domain name, and a port number.

Click **Next** to continue the installation.

6. The Installation Path dialog is displayed. Specify the **destination directory** where the Viewer should be installed, or choose the **default installation destination directory**, then click **Next**.
7. The Account Information dialog is displayed. Here you define the Windows user account that will be used to run PAS.

During installation, when specifying a Windows user account for PAS, you **MUST** choose a user which has Administrator privileges. Otherwise, the installation will fail.
8. Once you have entered in all of the appropriate information, click **Install** to continue.
9. Once installation is done, click **Finish** to exit the Installer.

10. Go to [Check the Connection to PrizmDoc Server](#) to make sure PAS is correctly connected to PrizmDoc Server.

Windows Requirements & Supported Environments

PAS Windows Requirements & Supported Environments

PAS is only supported on 64-bit operating systems. PAS requires significant network throughput and disk I/O, but very little processing power.

Supported Operating Systems

NOTE: When using Windows Server 2016 or Windows Server 2019, you must use the Desktop Experience version. The core version doesn't contain the components needed for PrizmDoc.

- Windows Server 2012, 2012 R2
- Windows Server 2016, 2019 with Desktop Experience

Hardware Requirements

Requirements vary greatly based on usage and it is generally a good idea to find what best fits your expected usage. The [Sizing Guide](#) is a good place to start understanding what resources PAS uses and how to optimize them for your needs.

Unattended Install & Uninstall

Unattended Installation of PAS on Windows

PAS is delivered as part of the "PrizmDoc Client" installer. This installer can be run unattended.

NOTE: If you have an updated license, you must re-start PAS and PrizmDoc Server in order to use the new license.

There are two required properties:

- `ServiceUser` - **Required.** Windows account which the PAS service should be run with. Must be in the format `DOMAIN\USER`.
- `ServicePassword` - **Required.** Password for `ServiceUser`.

There are several optional properties:

- `InstallFolder` - Base installation directory for the product. Default is `"C:\Prizm"`.
- `SelectedClientFeatures` - Features to install. Can be either empty or a comma-separated list of values containing any of the following:
 - `"HTML5Viewer"`
 - `"LocalFileViewerFeature"`
- `IISConfigureSamples` - Whether to configure the samples with IIS or not. Set to `"1"` for yes and set to an empty string for no. Default is `"1"`.
- `IISReregister` - Whether to re-register ASP.NET v4 with IIS or not. Set to `"1"` for yes and set to an empty string for no. Default is `"1"`.
- `SelectedPASFeatures` - Can be set to `"ALL"` to include PAS features or set to an empty string to not include them. Default is `"ALL"`.

Finally, you can also specify how PAS should connect to PrizmDoc Server. If you do, you must provide all three of the following properties:

- `PrizmScheme` - PrizmDoc Server protocol. Value may be either `"HTTP"` or `"HTTPS"`.
- `PrizmHost` - PrizmDoc Server hostname or IP address.
- `PrizmPort` - PrizmDoc Server port.

Installing with Traditional Linux Install Packages (deprecated)

DEPRECATION NOTICE: While we currently continue to offer and support traditional Linux packages for direct installation on a Linux host, these have largely become obsolete now that Docker deployment is an option. **We have announced deprecation of our traditional Linux install packages** and, in a future product release, we intend to only offer our [Docker-based deployment option](#). The rest of this topic applies to traditional Linux install packages only.

PAS is delivered as part of the "PrizmDoc Client" installation package (available as deb, rpm, and generic Linux).

Some steps are specific to a particular Linux distribution; these steps will be labeled as being specific to one of the following:

- Red Hat / CentOS Linux Distributions
- Ubuntu Linux Distributions

If you have questions about requirements before installing, refer to the [System Requirements & Supported Environments](#) topic.

NOTE: Make sure you log in as **root** to the machine.

NOTE: If you have an updated license, you must re-start PAS and PrizmDoc Server in order to use the new license.

Step 1 - Download the "PrizmDoc Client" Installation Package Appropriate for Your Linux Distribution

NOTE: Packages are only available for 64-bit systems.

1. Download **PrizmDoc Client** from the [website](#) by selecting the desired Linux Distribution.
OR
2. Download directly to the Linux server using the 'wget' command for the specific distribution as shown below:

NOTES:

1. **You must substitute the version of the package you are using in the code examples below.** For example, if you are using v13.8, then specify "13.8" instead of "**<version>**". If the version is a hot fix, you will also need to specify the hot fix number, for example, "13.8.1".
2. **Instructions assume that one already has 'wget' installed on the server OS.**

Red Hat Enterprise Linux / CentOS v7 and Later

```
wget
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_client_<version>.x86_64.rpm.tar.gz
```

Ubuntu

```
wget
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_client_<version>.amd64.deb.tar.gz
```


Generic .tar.gz

```
wget
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_client_<version>.x86_64.tar.gz
```

Step 2 - Unpack & Install the Downloaded Archive

Open a command line and change to the location where you downloaded the tarball. Use the following command line examples appropriate for your distribution to:

1. Decompress and unpack the downloaded file. After you have unpacked the archive, the contents will have been decompressed into a directory named **prizmdoc_client_<version>.<arch>[.rpm|.deb]**.
2. Change to the unpacked directory and install the packages.

Red Hat, CentOS, and Older Linux Distributions

```
tar -xzvf prizmdoc_client_*.rpm.tar.gz
cd prizmdoc_client_*.rpm
yum install --nogpgcheck *.rpm
```

Ubuntu

```
tar -xzvf prizmdoc_client_*.deb.tar.gz
cd prizmdoc_client_*.deb
sudo dpkg -i *.deb
# 'dpkg' does not resolve dependencies automatically, so please ignore possible
errors, if you did not install required dependencies yet, and invoke next commands
sudo apt-get update
sudo apt-get -f install
```

Generic .tar.gz Distribution

We also provide a generic .tar.gz package. Please review the [System Requirements and Supported Environments](#) topic to ensure compatibility. You will also need to install the **dependencies** described in the [Requirements](#) section. Once the dependencies are installed, you can install the **.tar.gz** with the following commands as root:

```
tar -xzvf prizmdoc_client_*.tar.gz
cd prizmdoc_client_*
ls prizm-*.tar.gz | xargs -n1 tar xzf
cp -R prizm /usr/share/
```

3. Go to the [Samples](#) topic for instructions on how to configure the connection between:
 - o Your Viewer web tier and PAS and,
 - o between PAS and PrizmDoc Server.

Step 3 (Optional) - Add PAS to the Boot Sequence

You can configure PAS to start/stop together with the system in two steps:

1. Create a symbolic link to `./pas/pm2/pas.sh` in the `/etc/init.d/` directory:

```
ln -s /usr/share/prizm/pas/pm2/pas.sh /etc/init.d/pas
```

2. Register PAS as an init script, so that it is managed by the system. This step is platform-dependent.

Red Hat, Fedora, CentOS, and Older Linux Distributions

```
chkconfig --add pas
```

Ubuntu

```
update-rc.d pas defaults
```

Once this is done, the system should stop PAS when going to reboot or shutdown, and will be started again when booting the server.

Excluding PAS from the Boot Sequence

If you want to prevent PAS from starting/stopping together with the system, you need to revert Step 2 from the section above. This can be performed as follows:

Red Hat, Fedora, CentOS, and Older Linux Distributions

```
chkconfig --del pas
```

Ubuntu

```
update-rc.d -f pas remove
```

Requirements & Supported Environments for Traditional Linux Install Packages

DEPRECATION NOTICE: While we currently continue to offer and support traditional Linux packages for direct installation on a Linux host, these have largely become obsolete now that Docker deployment is an option. **We have announced deprecation of our traditional Linux install packages** and, in a future product release, we intend to only offer our [Docker-based deployment option](#). The rest of this topic applies to traditional Linux install packages only.

PAS requires significant network throughput and disk I/O, but very little processing power.

Hardware Requirements

Requirements vary greatly based on usage and it is generally a good idea to find what best fits your expected usage.

The [Sizing Guide](#) is a good place to start understanding what resources PAS uses and how to optimize them for your needs.

Supported Linux Distributions

64-bit editions of:

- CentOS 7
- Red Hat Enterprise Linux 7
- Ubuntu 18.04 LTS

Uninstall Traditional Linux Install Packages

NOTE: This topic applies to traditional Linux install packages only.

To uninstall PAS from your Linux system, perform the following steps:

Make sure you log in as **root** to the machine.

1. Stop the service, depending upon where you installed. This will usually be:

```
/usr/share/prizm/pas/pm2/pas.sh stop
```

2. Next remove the installed files:

Ubuntu:

```
apt-get remove prizm-contentconnect prizm-pas
```

Red Hat/CentOS:

```
yum remove prizm-contentconnect prizm-pas
```

Generic Package:

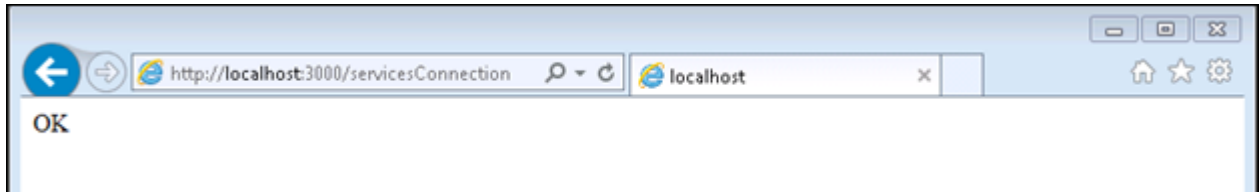
```
rm -rf /usr/share/prizm/LocalFileViewer
rm -rf /usr/share/prizm/Samples/Documents
rm -rf /usr/share/prizm/Samples/FormDefinitions
rm -rf /usr/share/prizm/Samples/imageStamp
rm -rf /usr/share/prizm/Samples/jsp
rm -rf /usr/share/prizm/Samples/markup
rm -rf /usr/share/prizm/Samples/markupLayerRecords
rm -rf /usr/share/prizm/Samples/viewingPackages
rm -rf /usr/share/prizm/viewer
rm -rf /usr/share/prizm/pas
```

NOTE: There may be temporary files left behind, like log and cache files, because they are not part of the installation packages. You may leave the temporary files or review them before deleting them.

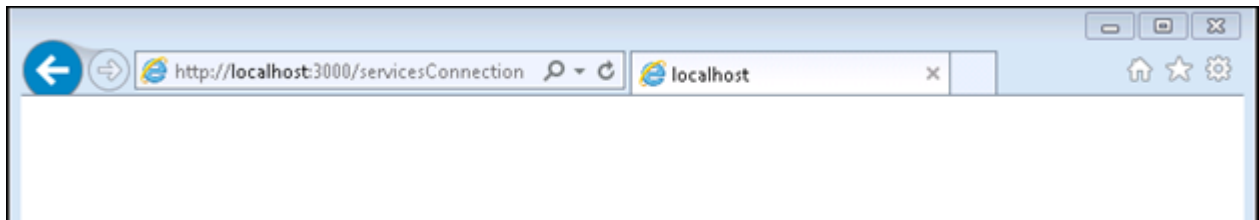
Check the Connection to PrizmDoc Server

PAS has a simple URL that you can use to verify the connection to PrizmDoc Server:

1. On the machine where you installed PAS, open your web browser and navigate to `http://localhost:3000/servicesConnection`. If you have changed the default port from **3000**, use the correct port for your instance of PAS.
2. If the connection is active, you will see **OK** on the page:



If the connection is not available, you will see a blank screen:



To troubleshoot connection issues between the Viewer and the backend, look at one of the following sections, depending on whether you have a [self-hosted backend](#) or [PrizmDoc Cloud backend](#).

Troubleshoot Connection Issues to Self-Hosted Backend

If there is an issue with your connection, there are a few steps you can take to troubleshoot:

1. Locate your **PrizmDoc Application Services (PAS) config file**. Assuming the standard install location would be `C:\Prizm\pas\pcc.win.yml` on Windows and `/usr/share/prizm/pas/pcc.nix.yml` on Linux.
2. Look at the values set for the `pccServer.hostName`, `pccServer.port`, and `pccServer.scheme` to confirm they are correct:
 - `pccServer.hostName` should specify the machine where you installed PrizmDoc Server.
 - `pccServer.port` should specify the port that PrizmDoc Server is using, which is `18681` by default for both single-server mode and [cluster mode](#). If the settings are not correct, update them and try again.
3. If the values for these parameters are correct, try restarting **PAS**. Refer to [Starting & Stopping PAS](#) for instructions, and try again.

If you are still unable to verify your connection to PrizmDoc Server, please contact [Accusoft Support](#).

Troubleshoot Connection Issues to PrizmDoc Cloud Backend

If there is an issue with your connection, there are a few steps you can take to troubleshoot:

1. Locate your **PrizmDoc Application Services (PAS) config file**. Assuming the standard install location, this would be `C:\Prizm\pas\pcc.win.yml` on Windows and `/usr/share/prizm/pas/pcc.nix.yml` on Linux.
2. Look at the values set for the `pccServer.hostName`, `pccServer.port`, and `pccServer.scheme` to ensure the

following (the `pccServer.apiKey` should be set to your API key):

- o `pccServer.hostName`: "api.accusoft.com"
- o `pccServer.port`: 443
- o `pccServer.scheme`: "https"

3. If the values for these parameters are correct, try restarting **PAS**. Refer to [Starting & Stopping PAS](#) for instructions, and try again.

If you are still unable to verify your connection to the PrizmDoc Cloud, please contact [Accusoft Support](#).

Configuring

This section covers common configuration options that you, as a PrizmDoc Viewer admin, will want to consider for PAS:

- [PAS Configuration](#) - Configuration options for finding documents, storing logs, and connecting to a database.
- [PAS Database Administration & Maintenance](#) - Information on configuring PAS to communicate with your database.
- [How & When to use CORS](#) - Steps for configuring PAS to enable CORS.

PAS Configuration

PrizmDoc Application Services Configuration

The PrizmDoc Application Services (PAS) use a central configuration file to determine, among other things, where to find documents, where to store logs and how to connect to a database.

Note that PrizmDoc Cloud uses a specific default value that differs from the self-hosted default value: `defaults.viewingSessionTimeout` is 5h. This is the only value that differs between PrizmDoc Cloud and self-hosted for PAS configuration.

Configuration File Location

On Windows, assuming a default installation, the configuration file is located at `C:\Prizm\pas\pcc.win.yml`.

On Linux, assuming a default installation, the configuration file is located at `/usr/share/prizm/pas/pcc.nix.yml`.

Default Configuration

Among other things, the config file includes the `port`, `secretKey`, `logs.path`, `defaults.viewingSessionTimeout`, and `defaults.viewingPackageLifetime` properties.

`port` defines the port that PAS will use to listen to its HTTP connection.

`secretKey` defines a shared secret value which must be provided in requests to critical [PAS REST API](#) routes (typically non-GET routes that modify data). If you host your PAS server on the public internet, callers must provide this value via an `Accusoft-Secret` request header in order for many of the REST API routes to function. **To keep your application as secure as possible, we strongly recommend you change the default value of this property to a unique string that only your application uses.**

`logs.path` determines the location on the local filesystem where the logs for PAS will be stored.

`defaults.viewingSessionTimeout` is the length of time that a viewing session remains usable. For example, "20m" indicates viewing sessions will timeout after 20 minutes.

`defaults.viewingPackageLifetime` is the minimum time for the created viewing package content to remain available. If missing, the value is considered to be 24 hours. If set to 0, viewing package content will remain available perpetually. This value could be overridden by the `input.viewingPackageLifetime` property in individual `viewingPackageCreators` requests when starting a new viewing package creator process. This is a beta feature that is not officially supported by Accusoft and its behavior can be changed at any time in a future version of the product.

The value of the above two properties must be an integer, followed by "s", "m", "h", or "d". The suffixes stand for second, minute, hour, or day, respectively. There should not be any space characters between the number and suffix.

NOTE: For PrizmDoc Cloud, the default value for `defaults.viewingSessionTimeout` is 5h.

Configuring the PrizmDoc Server Connection

The connection to your desired PrizmDoc Server, whether Self-Hosted or Cloud, can be configured through the `pccServer` object, which has the following properties:

- `pccServer.hostName` - the hostname to use to connect to PrizmDoc Server.
- `pccServer.port` - the port on the above hostname to connect to.
- `pccServer.scheme` - the scheme to use to communicate with PrizmDoc Server. This can be set to `http` or `https`.
- `pccServer.apiKey` - the API Key that PAS should use if you are using the PrizmDoc Cloud Server. This value will be ignored for a PrizmDoc Self-Hosted Server.

Configuring Storage

There are various data entities stored or accessed by PAS, as noted by comments in the config file. All of these can be configured separately.

Each storage entity will have a name, such as `documents` or `markupLayerRecords`, and each named entity will have a `.storage` property, such as `documents.storage`. This property defines the kind of storage that will be used. The supported values are as follows:

- `"filesystem"` - Store on the local filesystem or network attached storage that has been mapped to a local drive or folder. For any data entity configured to be stored on the filesystem, the following additional properties are required:
 - `.path` - Folder location where the data should be stored. On Windows, this can also include environment variables. If these paths are changed from the default values, PrizmDoc must be granted write permissions for them to function.
- `"database"` - Store inside the configured database. See [Configuring the database](#) below for more details.
- `"s3"` - Store files in S3 Buckets. For any data entity configured to be stored in S3 Buckets, the following additional properties are required:
 - `.s3BucketName` - Name of the bucket that you would like to store the specified data entity type. This bucket must be unique across all of Amazon S3 services and must be created in your S3 dashboard.
 - `.path` - Base "key" the data should be organized into. S3 uses keys as its directory structure. This must have no starting / and no trailing /.
- `"azureBlobStorage"` - This is a beta feature that stores files in Microsoft Azure Blob Storage. For any data entity configured to be stored in Azure blob storage, the following additional properties are required:
 - `.azureBlobStorageConnectionString` - Connection string to the Microsoft Azure Blob

Storage.

- `.azureBlobStorageContainerName` - Name of the container in the Microsoft Azure Blob Storage that you would like to store the specified data entity type.
- `.path` - Sub-key of the `.azureBlobStorageContainerName` where the data should be stored.

The following table shows the storage entities and supported storage providers:

Storage Entity	Storage Type
documents	filesystem, s3, azureBlobStorage
markupXml	filesystem, s3, azureBlobStorage
markupLayerRecords	filesystem, s3, azureBlobStorage
formDefinitions	filesystem, s3, azureBlobStorage
imageStamps	filesystem, s3, azureBlobStorage
viewingPackagesData	database
viewingPackagesProcesses	database
viewingPackagesArtifacts	filesystem, s3, azureBlobStorage
viewingPackagesArtifactsMetadata	database
viewingSessionsData	database
viewingSessionsProcessesMetadata	database

NOTE: Not all storage entities are compatible with all storage providers. Checking for these values is done on start up and an informative error will be logged to the PAS log, `{installDir}/logs/pas`, in the case of a mismatch.

Additional S3 Notes:

You must handle Amazon Web Services credentials in one of the available ways that are provided on the following pages:

Loading credentials in Node.js from IAM Roles for EC2:

<http://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/loading-node-credentials-iam.html>

Loading credentials from a shared credentials file:

<http://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/loading-node-credentials-shared.html>

Loading credentials from environment variables:

<http://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/loading-node-credentials-environment.html>

Examples

```
documents.storage: "filesystem"
documents.path: "/usr/share/prizm/Samples/Documents"
```

```
documents.storage: "filesystem"  
documents.path: "C:\\Docs\\Docs"
```

```
documents.storage: "filesystem"  
documents.path: "\\servername\sharename\docs\docs"
```

```
markupLayerRecords.storage: "filesystem"  
markupLayerRecords.path: "%ALLUSERSPROFILE%\Accusoft\Prizm\MarkupLayerRecords"
```

```
viewingPackagesData.storage: "database"
```

```
documents.storage: "s3"  
documents.s3BucketName: "myS3Bucket"  
documents.path: "Samples/Documents"
```

```
documents.storage: "azureBlobStorage"  
documents.azureBlobStorageConnectionString: "AccountName=..."  
documents.azureBlobStorageContainerName: "pas"  
documents.path: "Documents"
```

NOTE: some data entities have limitations on the kind of storage that they can be stored in. If PAS is misconfigured, it will not start correctly. It's best to keep a copy of the defaults so that you can revert them if you need to.

Legacy Mode

Legacy Mode refers to being able to open markup files that were created using one of the Web Tier Samples available before the release of PAS. To work correctly, it needs to be enabled on `documents`, `markupXml`, and `markupLayerRecords`, as such:

```
documents.legacyMode: true  
markupXml.legacyMode: true  
markupLayerRecords.legacyMode: true
```

If you do not need this feature -- for example, if PAS is the first time you are using markup files -- you can turn this off by setting all values to `false`, which will provide a small performance gain when working with the markup APIs. Note that all markup files created by PAS itself, regardless of whether `legacyMode` was on or off, will be compatible with PAS when `legacyMode` is off.

Feature Toggles

Some features in PAS are behind feature flags, and they can be turned on or off. This is done through the `feature.*` options in the config file. The values can be set to:

- `enabled` - turns the feature on
- `disabled` - turns the feature off

You can also remove the specific feature configuration value altogether in order to observe the default behavior for that feature. The list of features is:

- `viewingPackages` - default: `disabled` - Enables Pre-Conversion Services and APIs, which allow you to pre-convert documents and cache on-demand document views in PAS, improving the speed at which documents can be viewed, as well as reducing the processing time in PrizmDoc Server for repeat document views.

Example

```
feature.viewingPackages: enabled
```

Configuring the Database

IMPORTANT: A database is required in order to use Viewing Packages. This feature is disabled by default and will need to be turned on. Without turning Viewing Packages on, PAS will not use a database, or even check for its existence in the configuration.

PAS requires configuration to a database, allowing it to store some of its data there. It will not start correctly without having a correctly configured and accessible database. The following config properties are available in PAS to support that:

- `database.adapter` - the type of database being used. The following values are supported:
 - `sqlserver` - Microsoft SQL Server
 - `mysql` - MySQL
- `database.connectionString` - the connection string to the database.
 - For Microsoft SQL Server, you can use either a classic connection string or a URI-based connection string.
 - For MySQL, you must use a URI-based connection string.

Microsoft SQL Server Examples

```
database.adapter: sqlserver
database.connectionString: "Server=tcp:prizm-
server.database.host,1433;Database=prizmdb;User ID=prizm-
user;Password=password;Encrypt=True"
```

```
database.adapter: sqlserver
database.connectionString: "mssql://prizm-user:password@prizm-
server.database.host:1433/prizmdb?encrypt=true"
```

MySQL Example

```
database.adapter: mysql
database.connectionString: "mysql://prizm-user:password@prizm-
server.database.host:3306/prizmdb"
```

Alternately, instead of specifying a `database.connectionString`, you can use the following older options to connect to the database:

- `database.host` - the hostname to use to communicate with the database.
- `database.port` - the port to use to communicate with the database.
- `database.user` - the user to use when connecting to the database.
- `database.password` - the password for the specified user.
- `database.database` - the database name to use on the database server.

But we strongly recommend you use `database.connectionString` instead as it is more flexible.

Configuring Cross-Origin Resource Sharing

While you can set up CORS quite easily through your web server, PAS also supports settings CORS headers directly. It is exposed through the `cors` config object, as such:

- `cors.enabled` - whether to set CORS headers. Supported values are `true` or `false`.
- `cors.allowedOrigins` - an array of the exact origin values to support. Not including this value will cause all CORS requests to be denied. This array will be used to determine the `Access-Control-Allow-Origin` header.
- `cors.exposedHeaders` (optional) - an array of headers keys to allow the browser to read. This value is configured to include headers returned by PAS by default.

Example

```
cors.enabled: true
cors.allowedOrigins: [ "http://example.com", "https://example.com" ]
```

PAS Database Administration & Maintenance

Introduction

A database must be provided to PAS in order to use the Pre-Conversion Services feature. You can see a list of supported databases in the [PAS Configuration](#) topic. After configuring PAS with the correct information, some databases, like Microsoft SQL Server, will require that a script is run in order to set up the correct tables for that database. You can find out more information about this in the topic for [setting up your database](#). Please note that PAS itself will only require read/write access to the database; running the mentioned scripts will require access to create and migrate tables.

Maintaining the Database

While directly reading, linking, or otherwise using the data stored in the database by PAS is discouraged, you will still need to do regular administrative tasks, such as taking proper snapshots and backups of the data in order to prevent and mitigate data loss.

In the event of data loss that requires recovery from a backup (both for the database or the local file storage) PAS has an API to validate viewing packages and their state. For more information, refer to the [Viewing Packages API](#).

Product and Database Updates

As the Pre-Conversion (Viewing Packages) feature is updated in future releases, the product will contain the necessary logic or scripts to transition existing tables and data to the new format, if a schema change is necessary. You can find out more information on updating PAS in the topic for [setting up your database](#).

How & When to use CORS

Do you need to use CORS?

We actually recommend an application design that **does not** require CORS.

While it is true that the Viewer, running in the browser, needs a way to send requests to PAS, we recommend you set up a dedicated reverse proxy route as part of your web application (or web server) which allows the Viewer to make requests to the same origin (domain) from which the web page itself was loaded. Our [Getting Started section](#) explains this sort of deployment in a lot more detail. We recommend you read through that first if you have not already.

That said, if you want your viewer instances in the browser to send requests directly to your PAS deployment running on a different domain (origin), it is possible. To do this, you'll need to explicitly configure PAS to enable CORS:

1. Modify the **CORS settings** within your PAS configuration file. Refer to the topic, [PAS Configuration](#), for more information.
2. Restart **PAS** for the changes to take effect. Refer to the topic, [Starting & Stopping PAS](#), for more information.
3. Update the **imageHandlerUrl** in the viewer initialization options to point directly to the publicly accessible PAS entry point. For more information on viewer configuration options, refer to the topic, [Initialization Parameters](#).

NOTE: *There is no native support for CORS in Internet Explorer 8 and 9.*

Clustering

This section contains the following information:

- [Optimize Cache Performance for Cluster Environments](#)
- [Run PAS on Clusters](#)

Optimize Cache Performance for Cluster Environments

Introduction

PrizmDoc Application Services (PAS) utilizes cache locations that don't need to be on the same server. If multiple PAS servers are in use, they can be configured to use the same central filesystem and database so that cached data is shared between servers.

A given Viewing Session is cached by either PrizmDoc Server or by PAS depending on how it was created. Sessions created using a documentId are stored in PAS' central cache as a [pre-converted viewing package](#). As these sessions are not cached in PrizmDoc Server, they will not be directly accessible through the PrizmDoc Server API.

Example

```
POST http://localhost:3000/ViewingSession Content-Type: application/json
{
  "source": {
    "type": "document",
    "fileName": "sample.doc"
    "documentId": "doc_9495837910qc"
  }
}
```

Viewing packages can be created either explicitly via the [Pre-Conversion API](#) or, as in the above example, implicitly by providing a documentId in the [Viewing Session API](#).

Sessions created without a documentId are not stored as viewing packages and are cached by PrizmDoc Server as normal according to the server configuration and request parameters.

Example

```
POST http://localhost:3000/ViewingSession Content-Type: application/json
{
  "source": {
    "type": "document",
    "fileName": "sample.doc"
  }
}
```

NOTE: If PrizmDoc Server is running in Cluster Mode, PAS handles the use of affinity hints internally and does not require the user to perform anything additional for optimized PrizmDoc Server cache performance.

Run PAS on Clusters

Introduction

PrizmDoc Application Services (PAS) is designed to scale out well. By default, it will run multiple threads on a single machine whenever it is capable, and it can be installed to run on multiple machines easily. You will need to do the following in order to run PAS on multiple servers:

- Install the PAS component on all machines that you would like to use.
- On each machine, configure all filesystem-based storage in the PAS configuration file to point to a shared location. It is recommended that you use a shared Network Attached Storage (NAS) device that is accessible to all PAS instances.
- On each machine, [find the correct PAS configuration file](#), and adjust the settings so that each instance of PAS is pointing to the same PrizmDoc Server or PrizmDoc Cloud entry point.

You can find out more about configuring PrizmDoc Server with a cloud entry point for a cluster environment in the [PrizmDoc Viewer Cluster Mode](#) topic.

Make sure to re-start PAS after every configuration change, in order for the changes to take effect.

Load Balancing

There is no special consideration for load balancing several PAS servers. Any request can be routed to any instance of PAS, and you can use any off-the-shelf load balancer to handle the routing.

Starting & Stopping

This section contains the following information:

- [Linux](#)
- [Windows](#)

Linux

Introduction

The included script `./pas/pm2/pas.sh` can be used to start and stop the PrizmDoc Application Services (PAS). The following examples assume the default install location. If you did not install it to the default location, replace `/usr/share/prizm` with the location to which you installed it.

Start

Example

```
/usr/share/prizm/pas/pm2/pas.sh start
```

Stop

Example

```
/usr/share/prizm/pas/pm2/pas.sh stop
```

TIP: If you created a symbolic link to `./pas/pm2/pas.sh` in the `/etc/init.d/` directory, you can use the `service` command to manage PAS instead of invoking the script directly. That way if you don't need PAS to automatically start/stop, you can complete the first step to more easily manage it.

The syntax is as follows:

Example

```
service pas start|stop|restart|status
```

Windows

Method 1: From the Windows Service

On Windows, the PrizmDoc Application Services (PAS) should ideally be started/stopped from the Windows service management console. As part of the PrizmDoc Viewer installation, the service should be configured to start automatically. If you need to start, stop, or restart, use the following instructions:

1. Log onto the system using an account with administrator privileges.
2. Click **Start > Run**.
3. Type **services.msc**.
4. Press **Enter**.
5. Find **PAS** in the list of services, and right-click on the service to access the context menu.
6. To Start the Service: Click **Start** and wait for the service to start. The status should update to "started" (this option will only be available if the service is not running).
7. To Stop the Service: Click **Stop** in the right-click menu and wait for the service control dialog. The status will be updated to be blank (this option will only be available if the service is already started).
8. To Restart the Service: Click **Restart** and wait for the service control dialog (this option will only be available if the service is already started).

Method 2: From the Command Line

If access to the control panel is not available, services can also be started/stopped from the command line using the following commands:

Example

```
net start "Prizm Application Services"  
net stop "Prizm Application Services"
```

Service Logs

The PAS Windows service will log certain status messages to the Windows Event Log. These messages can be helpful in diagnosing problems while starting and stopping the service. To view the Windows Event Log, use the following instructions:

1. Log onto the system using an account with administrator privileges.
2. Click **Start > Run**.
3. Type **eventvwr**.
4. Press **Enter**.
5. Expand **Applications and Services Logs**.
6. Click **PAS**.

Error Reporting

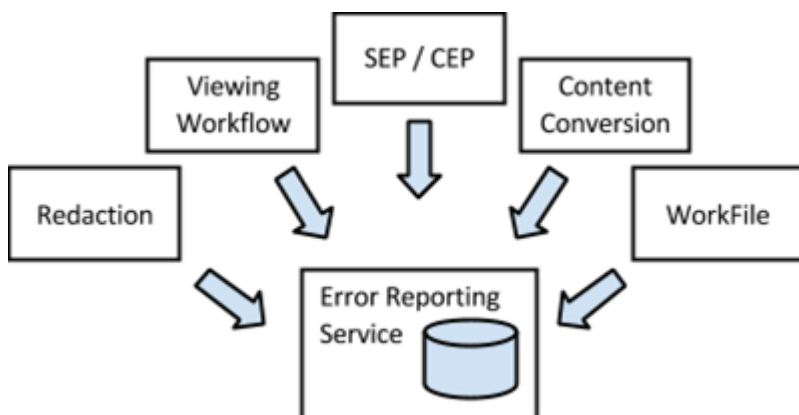
Introduction

This section contains the following information:

- [Accusoft Policy on Log Changes](#)
- [Search Tips](#)
- [Packaging Log Files for Product Support](#)

PrizmDoc Server is composed of a number of micro-services each responsible for some small processing task related to document conversion and viewing. Consequently, an error may occur while processing a small piece of a document and may not be readily apparent to the user.

While it is a simple task for one service to report an error to a calling service, it is not always appropriate to present that error information to the end user. The Error Reporting Service provides a centralized log into which all services report errors with the overall goal of simplifying system troubleshooting:



Error Reporting Configuration

Errors are reported to a log file located in the directory containing the other service log files by default:

- **Linux:** /usr/share/prizm/logs/PccErrors.log
- **Windows:** C:/Prizm/logs/PccErrors.log

These locations, as well as the number of daily logs to retain, can be found in the [central configuration](#) file. The file paths for the Central Configuration file are:

- **Linux:** /usr/share/prizm/prizm-services-config.yml
- **Windows:** C:\Prizm\prizm-services-config.yml

NOTE: The default installation directory is: C:\Prizm.

Error Report Entries

The error log file may contain two type of entries: Errors and Relations. Errors entries describe a specific error event reported by a service and can be used to diagnose issues with a document or service. Relations describe the relationship between two resources and are used to help diagnose an error in which the cause was the result of a failure from another related resource. These are described in further detail below.

Error Entries


An example Error entry is shown below:

Example

```
{
```

```

"gid": "duw97iCztVvreTmqRZdgOw",
"service": "ImagingServices",
"resourceType": "ViewingSession",
"resourceId": "
3eHY2FqlgNyHo3i2kx2zJQ38YvNpcQMG4CowwC_71cZ1jRN1l6k48PxBkPkAkGd0xWHXjWmkhdQoRw",
"relevance": 100,
"errorCode": "DocumentRequiresAPassword",
"time": "2015-04-28 20:27:25.0473",
"errorId": "f00hLsu_V8TimZm88w1b6w"
}
    
```

Error Entry	Description
gid	This is the Global ID assigned to each new request. It uniquely identifies operations resulting from the original request.
service	This is the name of the service which reported the error.
resourceType	This describes the resource related to the error. Examples include ContentConverter, WorkFile, MarkupBurner, RedactionCreator.
resourceId	This is the unique resource identifier (i.e. ViewingSessionId).
relevance	<p>This indicates the importance of the error:</p>  <ul style="list-style-type: none"> 100 A critical error from invalid input or lack of input that directly relates to a user problem. (i.e. missing password, missing file extension for txt file) An important error that likely directly relates to user problems. (i.e. unhealthy service, corrupt document) An error that may indirectly relate to a problem, but may not indicate the real cause. The real cause may be reported somewhere else. (i.e. intermediate page conversion failed) An error that may be related to a problem but isn't the direct cause. (i.e. Timeout waiting for some data) 0 An error that should have little, if any, impact on the user.
errorCode	This is the error code reported by the service. This will be a PascalCased string briefly describing the error.
time	This is the time at which the error was reported.
errorId	This is a unique errorId assigned to the error by the Error Reporting Service.

Relation Entries

An example relation entry is shown below:

Example

```
{
```



```

"gid": "duw97iCztVvreTmqRZdgOw",
"service": "ImagingServices",
"resourceType": "ViewingSession",
"resourceId": "3eHY2Fq1gNyHo3i2kx2zJQ38YvNPcQMG4CowwC_71cZ1jRN1DEF0UbuR16k48P",
"relation": "RedirectedViewingSession",
"relationResourceId":
"gRrJ8ay0jV6wBXiqMjxmB4epUrsd7KqVdtsD_BtwAZbhYBVpb4P2ksm0_kcByugmA",
"relationId": "EjA6CpDhFuTMP3sTAvMhyA"
}

```

Relation Entry	Description
gid	This is the Global ID assigned to each new request. It uniquely identifies operations resulting from the original request.
service	This is the name of the service which reported the relation.
resourceType	This describes the resource associated with the relation. Examples include ContentConverter, WorkFile, MarkupBurner, RedactionCreator.
resourceId	This is the unique resource identifier (i.e. ViewingSessionId).
relation	This describes the related resource. (i.e. RedirectedViewingSession).
relationResourceId	This is the unique related resource identifier (i.e. ViewingSessionId).
relationId	This is a unique relationId assigned to the error by the Error Reporting Service.

Accusoft Policy on Log Changes

Introduction

Accusoft is moving towards using JSON as the format for each log line. This makes it easier for us to programmatically analyze log data to improve the product. While it is possible to write your own programs that analyze our log data, please be aware that raw log data may change from release to release, and that logged events may be added, removed, or changed in a variety of ways. Specifically:

- New log events may be added.
- New properties may be added to existing log events.
- Existing log events may no longer be logged.
- Existing properties may be removed from existing log events.
- The minimum level of an existing log event (10 for TRACE, 20 for DEBUG, 30 for INFO, etc.) may change.

Additionally, log files that do not currently use JSON as their logging format will be replaced at some point by JSON logging.

Search Tips

Introduction

The examples below show methods for locating errors in the Error Reporting Service logs. While the examples here use command-line based searches, the same results can be achieved with your favorite text editor or other search tool.

Searching for a Specific Relevance

Errors that can be resolved directly by a user will have a high relevance (typically 90 or 100). It is useful to filter the PccErrors.log for entries with a specific relevance. The examples below show command line operations to list errors with relevance 100 on both Windows and the Linux Bash shell.

On Windows

```
C:> findstr /L "relevance":100 C:\ProgramData\Accusoft\Prizm\Logs\PccErrors.log
```

On Linux Bash

```
$> grep 'relevance':100 /usr/share/prizm/logs/PccErrors.log
```

An example of an entry with relevance 100 is shown below:

Example

```
{ "gid": "13CbzVdHZ/wqVKLCZmfq9A", "time": "2015-05-14 19:26:25.9842", "resourceType": "ViewingSession", "resourceId": "190054e9-574f-4b83-ae86-d30c0c7e2c1c", "relevance": 100, "errorCode": "DocumentRequiresAPassword", "service": "ImagingServices", "errorId": "3SgxE5MseBEolbECEMSETA" }
```

In this case, the user has attempted to view a document requiring a password without providing the password.

Searching for a Viewing Session

If an error occurs during a particular viewing session, the cause of the problem may be reported in the Error log. The errors for a particular viewing session can be found first by searching for the viewing session ID.

On Windows

```
C:> findstr /L "sugq5PRxDV4ERAbQLgpRzKjbKrHp868m6zN2QG-wBFHO3ZX-SihJ3GIJA8FK4WZzB2Djij_gZWTAUdyKeqBcw" C:\ProgramData\Accusoft\Prizm\Logs\PccErrors.log
```

On Linux Bash

```
$> grep sugq5PRxDV4ERAbQLgpRzKjbKrHp868m6zN2QG-wBFHO3ZX-SihJ3GIJA8FK4WZzB2Djij_gZWTAUdyKeqBcw /usr/share/prizm/logs/PccErrors.log
```

Example results from the query are shown below:

Example

```
{ "gid": "13CbzVdHZ/wqVKLCZmfq9A", "time": "2015-05-14 19:26:25.9834", "resourceType": "ViewingSession", "resourceId": "sugq5PRxDV4ERAbQLgpRzKjbKrHp868m6zN2QG-wBFHO3ZX-SihJ3GIJA8FK4WZzB2Djij_gZWTAUdyKeqBcw", "relation": "ViewingSessionId", "relationResourceId": "190054e9-574f-4b83-ae86-d30c0c7e2c1c", "service": "ImagingServices", "relationId": "HsayMuoWcABqBDsN6yR9Vg" }
```

Notice the second ID: "190054e9-574f-4b83-ae86-d30c0c7e2c1c". This is the internal ID for a viewing session. A search for this ID will locate any errors associated with the viewing session.

Searching for Related Resources

After a record has been identified, it is useful to determine if any related error records were reported. This can be achieved by searching the log for the resource ID.

On Windows

```
C:> findstr /L "190054e9-574f-4b83-ae86-d30c0c7e2c1c" C:\ProgramData\Accusoft\Prizm\Logs\PccErrors.log
```

On Linux Bash

```
$> grep 190054e9-574f-4b83-ae86-d30c0c7e2c1c /usr/share/prizm/logs/PccErrors.log
```

Example results from the query are shown below:

Example

```
{ "gid": "13CbzVdHZ/wqVKLCZmfq9A", "time": "2015-05-14
19:26:25.9842", "resourceType": "ViewingSession", "resourceId": "190054e9-574f-4b83-ae86-
d30c0c7e2c1c", "relevance": 100, "errorCode": "DocumentRequiresAPassword", "service": "ImagingServices", "errorId":
"3SgxE5MseBEo1bECEMSETA" }

{ "gid": "13CbzVdHZ/wqVKLCZmfq9A", "time": "2015-05-14
19:26:25.9834", "resourceType": "ViewingSession", "resourceId": "sugq5PRxDV4ERAbAQLGpRzKjbKrHp868m6zN2QG-
wBFH03ZX-
SIhJ3GIJA8FK4WzZB2DjiJ_gZWTAUdyKeqBcw", "relation": "ViewingSessionId", "relationResourceId": "190054e9-574f-
4b83-ae86-
d30c0c7e2c1c", "service": "ImagingServices", "relationId": "HsayMuoWcABqBDsN6yR9Vg" }

{ "gid": "13CbzVdHZ/wqVKLCZmfq9A", "time": "2015-05-14 19:26:25.9843", "resourceType":
"ViewingSession", "resourceId": "190054e9-574f-4b83-ae86-
d30c0c7e2c1c", "relation": "SourceDocumentWorkFile", "relationResourceId": "JAQ6o9ck1VM2ohFf5xiI6g", "service":
"ImagingServices", "relationId": "nXI4pNPhXms2Idb8vuilLQ" }
```

Notice the third record: "JAQ6o9ck1VM2ohFf5xiI6g". This reported relation record indicates the workfile requiring the password. Now that the workfile ID has been identified, it is possible to determine the actual file from the WorkfileService.log.

Searching Other Logs

Once the workfile ID is known, it possible to search WorkfileService.log for location of the actual file which caused the error.

On Windows

```
C:> findstr /L JAQ6o9ck1VM2ohFf5xiI6g C:\ProgramData\Accusoft\Prizm\Logs\WorkfileService.log
```

On Linux Bash

```
$> grep JAQ6o9ck1VM2ohFf5xiI6g /usr/share/prizm/logs/WorkfileService.log
```

Example results from the query are shown below:

Example

```
{ "name": "WorkfileService", "hostname": "ip-10-182-110-214", "pid": 7345, "taskId": 24003,
"gid": "Fa7Uay+IjZqa53pgSF9ueA", "level": 30, "type": "WorkfileRepository", "contentsName":
"/usr/share/prizm/cache/WorkfileCache/JAQ6o9ck1VM2ohFf5xiI6g/WorkfileContents.pdf", "msg":
"Creating content write stream", "time": "2015-05-14T19:26:25.872Z", "v": 0 }

{ "name": "WorkfileService", "hostname": "ip-10-182-110-214", "pid": 7345, "taskId": 24003,
"gid": "Fa7Uay+IjZqa53pgSF9ueA", "level": 30, "type": "WorkfileRepository", "workfile":
"/usr/share/prizm/cache/WorkfileCache/JAQ6o9ck1VM2ohFf5xiI6g/WorkfileContents.pdf", "msg":
"Workfile content created", "time": "2015-05-14T19:26:25.872Z", "v": 0 }

{ "name": "WorkfileService", "hostname": "ip-10-182-110-214", "pid": 7345, "taskId": 24003,
"gid": "Fa7Uay+IjZqa53pgSF9ueA", "level": 30, "reqBegin": true, "req": { "method": "POST", "path": "/FDS/detect",
"port": 38503, "data":
{ "src": "/usr/share/prizm/cache/WorkfileCache/JAQ6o9ck1VM2ohFf5xiI6g/WorkfileContents.pdf" } },
"msg": "", "time": "2015-05-14T19:26:25.872Z", "v": 0 }

{ "name": "WorkfileService", "hostname": "ip-10-182-110-214", "pid": 7345, "taskId": 24004,
"gid": "13CbzVdHZ/wqVKLCZmfq9A", "level": 30, "taskBegin": true, "parent": { "name": "PCCIS", "pid": 8130,
"taskId": 2519 }, "reqAccepted": true, "req":
{ "method": "GET", "path": "/PCCIS/V1/WorkFile/JAQ6o9ck1VM2ohFf5xiI6g",
"port": 19020 }, "msg": "", "time": "2015-05-14T19:26:25.908Z", "v": 0 }

{ "name": "WorkfileService", "hostname": "ip-10-182-110-
214", "pid": 7345, "taskId": 24004, "gid": "13CbzVdHZ/wqVKLCZmfq9A", "level": 30, "type": "WorkfileService",
"workfileId": "JAQ6o9ck1VM2ohFf5xiI6g", "msg": "Begin: getWorkfile", "time": "2015-05-14T19:26:25.908Z", "v": 0 }

{ "name": "WorkfileService", "hostname": "ip-10-182-110-
214", "pid": 7345, "taskId": 24004, "gid": "13CbzVdHZ/wqVKLCZmfq9A", "level": 30, "type":
"WorkfileService", "workfile": { "id": "JAQ6o9ck1VM2ohFf5xiI6g", "expirationDateTime": "2015-05-
15T19:26:25.871Z", "fileExtension": "pdf", "_version": 1, "cacheEnabled": false, "fileFormat": "pdf" },
"msg": "Workfile retrieved", "time": "2015-05-14T19:26:25.908Z", "v": 0 }
```

The following section in the record shows the location of the file:
/usr/share/prizm/cache/WorkfileCache/JAQ6o9ck1VM2ohFf5xiI6g/WorkfileContents.pdf".

Packaging Log Files for Support

Introduction

This section describes where to find the PrizmDoc Viewer log files and where to send them for support.

Linux

With a default installation of PrizmDoc Viewer, and with the logging settings left to their out-of-box defaults, you can find all of the log files in this directory:

```
/usr/share/prizm/logs/
```

If you need to send PrizmDoc Viewer log files to Accusoft Support, simply create a gzipped tarball containing everything in that directory.

NOTE: We encourage you to use Google Drive or Dropbox links. For additional options, please see the [Support Tickets](#) section of the Accusoft Customer Portal.

Windows

With a default installation of PrizmDoc Viewer, and with the logging settings left to their out-of-box defaults, you can find all of the log files in these two directories:

```
C:\Prizm\logs
```

```
%ALLUSERSPROFILE%\Accusoft\Prizm\Logs
```

If you need to send PrizmDoc Viewer log files to Accusoft Support, simply create one or more zip files containing everything in these two directories.

NOTE: We encourage you to use Google Drive or Dropbox links. For additional options, please see the [Support Tickets](#) section of the Accusoft Customer Portal.

API Reference

Introduction

PrizmDoc Viewer has powerful APIs that help you achieve the results you want for viewing, converting, OCR, annotating, redacting, e-signing, detecting forms, comparing documents, watermarking, and creating template-based forms.

Viewing

Client-Side APIs

- **Viewer Control** - The Viewer Control has a robust, client-side JavaScript API giving your application lots of ways to interact with the Viewer running in the browser. Use it to customize Viewer menus and features, programmatically control the Viewer, and respond to Viewer events.
- **E-Signature Controls** - The E-Signature Client API includes two kinds of UI controls: a form designer control (TemplateDesigner), and a form entry control (ESigner). You can use these controls and their APIs for implementing form entry in your application.

Server-Side APIs

- **PAS REST API** - The PAS REST API is what your web server should use to create a viewing session whenever you need to render a web page with a Viewer for an end user. It is also the API you should use if you want to pre-generate a viewing package. There are many additional viewing-related endpoints in this REST API which are used by the Viewer itself and which your server-side application is welcome to use as well. However, *if you need to do server-side document processing without a Viewer, PrizmDoc Server is a better fit. See "Backend Document Processing" below.*

Backend Document Processing

PrizmDoc Server

- **REST API** - The PrizmDoc Server REST API contains powerful functionality for server-side document processing, including conversion, redaction, annotation, watermarking, search, and OCR.
- **.NET SDK** - A wrapper around most of the PrizmDoc Server REST API, making it easy to use PrizmDoc Server functionality in .NET.

Cloud Authentication

- **Cloud Authentication** - All REST APIs to PrizmDoc Cloud (PAS and PrizmDoc Server) must be authenticated with an API key or OAuth token. This section explains how to do this.

Viewer Control

The Viewer API permits programmatic control over the Viewer.

Most API functionality is exposed by the ViewerControl - the core component of the Viewer. The Viewer UI/chrome builds off of the API members of the ViewerControl.

The Viewer API is **required** to:

- Modify the behavior of the Viewer
- Augment the behavior of the Viewer
- Build custom Viewer menus

The Viewer API is **not required** to:

- Customize the Viewer's layout or style
- Add or remove tabs
- Move or remove buttons and other inputs

You may want to review [Configuring the Viewer](#) for additional information. In addition, see the [API Examples](#), which demonstrate using the API to implement common scenarios.

This topic provides information about the following:

- [API Functionality](#)
- [Getting Started](#)

API Functionality

The Viewer API exposes the following functionality:

- Creating and destroying the Viewer
- Events
- Page navigation
- Zooming and fitting content
- Mark (annotation and redaction) CRUD
- Markup saving and loading
- Customizing mouse tools
- Searching document text
- Printing
- Getting page and document attributes

Getting Started

This section provides basic information for getting started using the API.

Access to the API

The API classes are exposed through the global `PCCViewer` object.

Example

```
window.PCCViewer;
```

A `ViewerControl` is created when instantiating the Viewer with the jQuery plugin, and it is accessible through the `viewerControl` property of the returned object.

Example

```
var viewer = $("#mydiv").pccViewer(options);
var viewerControl = viewer.viewerControl;
```

Viewer Ready

While initialization of the Viewer is being performed, it is unsafe to call some of the ViewerControl methods (these methods will throw an error during initialization). The `ViewerReady` event signals that initialization has completed, and it is safe to call all ViewerControl methods with valid data.

Example

```
var viewerControl = $("#mydiv").pccViewer(options).viewerControl;

// Subscribe to the "ViewerReady" event
viewerControl.on(PCCViewer.EventType.ViewerReady, function(ev) {
    // It is now safe to call all ViewerControl methods
    viewerControl.setCurrentMouseTool("AccusoftLineAnnotation");

    // It's also safe to use ECMA5 properties
    // (in supporting browsers).
    var currentPageNumber = viewerControl.pageNumber;

    // This is also a good time to subscribe to other events,
    // if you want to keep all API code in one place.
    viewerControl.on(PCCViewer.EventType.PageDisplayed, ...);
});
```

Page Count Ready

The ViewerControl will not get the page count of a document from PrizmDoc Server until after the `ViewerReady` event. Before the page count is returned, the ViewerControl assumes every document has one page.

The `PageCountReady` event signals when the ViewerControl has the actual page count from the server. Subscribe to this event to know when you can navigate to and interact with other pages in the document.

Example

```
var viewerControl = $("#mydiv").pccViewer(options).viewerControl;

// Subscribe to the "PageCountReady" event
viewerControl.on(PCCViewer.EventType.PageCountReady, function(ev) {
    var pageCount = ev.pageCount;

    // We can now navigate to other pages in the document.
    if (pageCount > 1) {
        // Go to the second page of the document.
        viewerControl.setPageNumber(2);
    }
});
```

It is safe to call Viewer navigation API members before the page count is available to the ViewerControl, but

navigation outside of the known page count (of one) is not possible.

EstimatedPageCountReady Event

Calculating the actual page count of some document formats can take a significant amount of time, and therefore waiting on an actual page count could make the Viewer appear unresponsive to the end user. To address this problem, the PrizmDoc Server may quickly generate an estimated page count.

The `EstimatedPageCountEvent` signals that the Viewer has an estimated page count and that navigation to other pages is possible.

Example

```
var viewerControl = $("#mydiv").pccViewer(options).viewerControl;

// Subscribe to the "PageCountReady" event
viewerControl.on(PCCViewer.EventType.EstimatedPageCountReady, ...);
```

- The estimated page count will not always be triggered.
- The estimated and actual page count can be different. If the estimated page count is greater than the actual page count, then for a short time it will be possible to navigate past the actual last page of the document. The Viewer will display a placeholder for these pages until the actual page count is available, when these placeholders will be removed.

Page Numbering

Page numbering in the API starts with 1. This is true for events and methods. The API does not support 0-based page indexes, so be careful when iterating over pages.

Example

```
// Navigate to the first page of the document
viewerControl.changeToFirstPage();

// An equivalent call would be
viewerControl.setPageNumber(1);
```

Pixels

In the API, the unit of measure for height, width, location, and other sizes is a pixel. To determine the height and width of a page in pixels, use the ViewerControl's `requestPageAttributes` method.

Example

```
// Get attributes of page 1.
viewerControl.requestPageAttributes(1).then(function(attributes) {
    var pageWidth = attributes.width;
    var pageHeight = attributes.height;

    // Add a rectangle that is the full width and height of page 1
```



```
viewerControl.addMark(1, "RectangleAnnotation")
    .setRectangle({x:0,
                  y:0,
                  height: pageHeight,
                  width: pageWidth});
});
```

It is advised to check page attributes before getting or setting the location of a [Mark object](#).

- For vector data provided by the server, a resolution of 72dpi is chosen, from which dimensions in pixels are calculated.
- It is not safe to assume that vector data will be provided by the server for a given document type.

Getters, Setters, and ECMA5 Properties

The API commonly uses getter and setter methods for accessing and modifying data associated with objects. Typically these methods are in pairs, unless the data is read-only.

Example

```
// read-write
viewerControl.getPageNumber(); // get the current page number
viewerControl.setPageNumber(1); // set the current page to 1

// read-only
viewerControl.getPageCount(); // get the page count
```

The API also offers ECMA5 Accessor properties that correspond to the getter and setters. These accessor properties offer a different style of coding, and they are only available in modern browsers that implement `Object.defineProperty`.

Example

```
if (Object.defineProperty) {
    viewerControl.pageNumber; // get the current page number
    viewerControl.pageNumber = 1; // set the current page to 1
    viewerControl.pageCount; // get the page count
}
```

Fluent Style: Method Chaining

The API is designed to support method chaining. Most methods will return the current context (object on which the method was called) in order to promote method chaining. Exceptions to this rule are getter methods and methods that complete asynchronously.

Example

```
// Method chaining when creating a rectangle annotation
viewerControl.addMark(1, "RectangleAnnotation")
```

```
.setRectangle({x: 0, y:0, width: 100, height: 100})
.setFillColors("#FFFFFF")
.setOpacity(127)
.setBorderThickness(6);

// Method chaining when manipulating the page
viewerControl
  .rotatePage(90)
  .zoomIn(1.2);
```

Note that using the ECMA5 accessor properties often discourages method chaining.

Promises

The API implements the Promises/A+ spec in the class `PCCViewer.Promise`.

Many methods that complete asynchronously return a `PCCViewer.Promise` object. Use the returned promise object to subscribe callbacks for the completion of the asynchronous event.

Example

```
// Use the promise to subscribe a callback to requestPageText
viewerControl.requestPageText(1).then(function(pageText) {
  alert("Text of page 1 is: " \+ pageText);
});
```

Two exceptions are `ViewerControl#print` and `ViewerControl#search`, which complete asynchronously but do not return a `PCCViewer.Promise` object. These methods both return objects that represent the requested print and search. Searching and printing have status events and are cancellable, for which they cannot be suitably represented with a promise object.

External: jQuery

jQuery

The jQuery Plugin namespace.

See: [jQuery Plugins](#)

Namespaces

fn

Namespace: fn

jQuery. fn

The jQuery Plugin namespace.

Methods

`pccViewer(optionsopt)` → {`PCCViewer.Viewer`}

Creates and embeds a new viewer in the first element of the set of matched elements.

Each call to this method will create and return a unique `ViewerControl` object. This will call `PCCViewer.Viewer#destroy` on any existing viewer embedded in the selected element.

If plugin options are provided, then a new viewer is created in the selected element and a `PCCViewer.Viewer` object is returned. This will call `PCCViewer.Viewer#destroy` on any viewer that already existed in the selected element.

If plugin options are not provided, then a viewer is not created. Instead, the `PCCViewer.Viewer` object associated with an existing viewer is returned.

Parameters:

Name	Type	Attributes	Description
options	<code>external:jQuery.fn~Options</code>	<optional>	Plugin options.

Returns:

Type

`PCCViewer.Viewer`

Example

```
//Note: these are already included in the PrizmDoc Samples
var pluginOptions = {
  documentID: viewingSessionId,      // documentID is a
  required property
  language: languageItems           // language is a
  required property
};

$(document).ready(function () {
  // Creates a new viewer in the div with id="viewer1"
  var viewer = $("#viewer1").pccViewer(pluginOptions);
});
```

```

    // Can also access the returned object through the
    plugin.
    var viewerA = $("#viewer1").pccViewer(); // Does not
    create a new viewer.
    viewerA === viewer;                       // true
  });

```

Type Definitions

DateFormat

The format to use when displaying a date. The table below outlines the supported date format tokens and provides example output.

	Token	Output
Month	M	1 2 ... 11 12
	MM	01 02 ... 11 12
Day	D	1 2 ... 30 31
	DD	01 02 ... 30 31
Year	YY	70 71 ... 29 30
	YYYY	1970 1971 ... 2029 2030
Hour	H	0 1 ... 22 23
	HH	00 01 ... 22 23
	h	1 2 ... 11 12
	hh	01 02 ... 11 12
Minute	m	0 1 ... 58 59
	mm	00 01 ... 58 59
AM/PM	A	AM PM
	a	am pm

Type:

- String

LanguageOptions

This object is the contents of the `language.json` file present in all of the HTML5 viewer samples. This file is generally read server-side and passed into the jQuery plugin. It is strongly encouraged to keep a copy of the original file before doing any edits or translations. This object is required by the viewer UI.

For more information on this language file or localization, please consult the help section titled "Localizing the Viewer".

Type:

- Object

Options

The options object used for the HTML5 viewer jQuery Plugin, [external:jQuery.fn#pccViewer](#). This object is a superset of the main ViewerControl options, [PCCViewer.ViewerControl~ViewerControlOptions](#). All available options for the ViewerControl are also valid here, and will be passed as-is to the ViewerControl during initialization. The following will only include options specific to the jQuery plugin and the Viewer UI functionality inside the `viewer.js` file.

Type:

- Object

Properties:

Name	Attributes	Description
<code>documentID</code> : string		The ID of the document to load. This is the <code>documentID</code> property of the PCCViewer.ViewerControl~ViewerControlOptions options object.
<code>documentDisplayName</code> : string	<optional> Default: "file"	A meaningful name for the document used when downloading a signed document. See PCCViewer.ViewerControl~ViewerControlOptions for more information. Example: "sample.doc"
<code>imageHandlerUrl</code> : string	<optional> Default: "../pcc.ashx"	The end point of the web tier serving the viewer. Unless specified, viewer.js running in the default .NET sample of the PCCViewer.ViewerControl~ViewerControlOptions options object.
<code>language</code> : external:jQuery.fn~LanguageOptions		Specifies the language to use for ViewerControl. Use this option to
<code>icons</code> : string		Specifies the SVG for the icons in the viewer. The symbol element in this SVG has a class name in the viewer HTML template for that symbol will be added to the matching class name. Note that the class name must start with "pcc-icon-".
<code>redactionReasons</code> : external:jQuery.fn~redactionReasons	<optional>	A list of reasons to be used for the redaction controls.
<code>annotationsMode</code> : string	<optional> Default: "LegacyAnnotations"	The annotationsMode specifies the mode in which annotations will be handled in the viewer. If the annotationsMode is set to "LegacyAnnotations" (default), all annotations will be handled as in releases prior to 10.3. The second

Name	Attributes	Description
		<p>"LayeredAnnotations". This option enables the layered annotations feature available in PrizmDoc Viewer v13.17 and higher. A convenience enumeration <code>AnnotationModeTypes</code> is available in the <code>viewer.js</code> file and a <code>viewer.AnnotationModeTypes</code> property is available in the <code>Viewer</code> object. This enumeration is part of the PrizmDoc Viewer API.</p> <p>Deprecation Notice: In the future, the "LegacyAnnotations" option will be deprecated.</p>
<code>annotationID : string</code>	<optional>	<p>Specifies the annotation file to be loaded by the viewer.</p> <p><i>This property is only observed when the "LegacyAnnotations" option is set to true.</i></p>
<code>attachmentViewingMode : string</code>	<optional> Default: "NewWindow"	<p>The "attachmentViewingMode" specifies the mode in which the attachments will be opened. The following options are available:</p> <ul style="list-style-type: none"> • "NewWindow": (default) The attachment is opened in the new browser window. The query parameter <code>?viewing={{ATTACHMENT_VIEWING_MODE}}</code> is added to the URL. • "ThisViewer": The document is opened in the current viewer. A change (along with its associated event) is triggered. Your client-side code needs to handle this change. You can subscribe to the <code>PCCViewer.EventType.ViewAttachment</code> event to know when this happens.
<code>autoLoadAnnotation : boolean</code>	<optional> Default: <code>false</code>	<p>If set to <code>true</code>, the specified annotation is loaded when the viewer launches.</p> <p><i>This property is only observed when the "LegacyAnnotations" option is set to true.</i></p>
<code>autoLoadAllLayers : boolean</code>	<optional> Default: <code>false</code>	<p>When set to <code>true</code>, all available layers are loaded into the document. This includes all layers specified in <code>requestMarkupLayerNames</code> and <code>getSavedMarkupNames</code>.</p> <p><i>This property is only observed when the "LayeredAnnotations" option is set to true.</i></p>
<code>editableMarkupLayerSource : string</code>	<optional>	<p>When set to "LayerRecordId", the layer with the record ID specified by the <code>editableMarkupLayerSource</code> will be loaded from JSON into the document. When set to "XmlName", the layer with the <code>editableMarkupLayerValue</code> specified in the <code>viewer</code> object will be loaded from XML into the document. When set to "LayerName", the layer with that name has been saved</p>

Name	Attributes	Description
		<p>case the layer will be loaded from "DefaultName", a new empty layer document but given the name specified by the <code>editableMarkupLayerValue</code> view.</p> <p><i>This property is only observed when <code>LayeredAnnotations</code> is set to <code>"LayeredAnnotations"</code> and <code>editableMarkupLayerValue</code> is set.</i></p>
<code>editableMarkupLayerValue</code> : string	<optional>	<p>When the <code>editableMarkupLayerSource</code> is set to <code>"LayerRecordId"</code>, this specifies the ID of the layer that will be loaded from the document. When the <code>editableMarkupLayerSource</code> parameter is set to <code>"XmlName"</code>, this specifies the name of the XML layer that will be loaded from the document, unless the XML layer view has been saved to JSON, in which case the layer will be loaded from JSON. When the <code>editableMarkupLayerSource</code> view is set to <code>"DefaultName"</code>, this specifies a new empty layer.</p> <p><i>This property is only observed when <code>LayeredAnnotations</code> is set to <code>"LayeredAnnotations"</code> and <code>editableMarkupLayerSource</code> is set.</i></p>
<code>lockEditableMarkupLayer</code> : boolean	<optional> Default: <code>false</code>	<p>When set to <code>true</code>, the buttons that open the annotations (for edit) dialog will be disabled and it is not possible to change the edit layer menu item that opens the edit layer dialog. The <code>lockEditableMarkupLayer</code> menu item will also be removed so it is not possible to enable the editable markup layer.</p>
<code>template</code> : Object		<p>This object holds the various templates used for the UI needs. The templates correspond to the templates available in all samples -- the file name as the template names here -- for example, <code>"Template.html"</code> to the end -- for example, <code>fileName</code> would correspond to <code>fileNameTemplate.html</code> file.</p> <p>Properties</p> <ul style="list-style-type: none"> • <code>viewer</code> : string <p>This is the main viewer template partial HTML and is parsed as a complete template variable.</p> <ul style="list-style-type: none"> • <code>contextMenu</code> : string <p>This is the template for the</p>

Name	Attributes	Description
		<p>edit annotations. It contains partial HTML and is parsed using Underscore template variables.</p> <ul style="list-style-type: none"> • <code>overwriteOverlay</code> : string <p>This is the template used for displaying when the Viewer detects that the document contains partial HTML and is parsed using Underscore to complete the template.</p> <ul style="list-style-type: none"> • <code>unsavedChangesOverlay</code> : string <p>This is the template used for displaying when the Viewer detects that the document is a markup file while having already loaded in the view HTML and is parsed using complete template variables.</p> <ul style="list-style-type: none"> • <code>printOverlay</code> : string <p>This is the template used for displaying when the user selects the print option. It contains partial HTML and is parsed using Underscore to complete the template.</p> <ul style="list-style-type: none"> • <code>print</code> : string <p>This is the template used for displaying when the user selects the print option. This template contains partial HTML and is parsed using complete template variables. Variables are parsed using Underscore. This template is part of the main application options, PCCViewer.ViewerControlOptions.</p> <ul style="list-style-type: none"> • <code>downloadOverlay</code> : string <p>This is the template used for displaying when the user selects the download option. It contains partial HTML and is parsed using Underscore to complete the template.</p> <ul style="list-style-type: none"> • <code>esignOverlay</code> : string <p>This is the template used for displaying when the user selects the esign option. It contains partial HTML and is parsed using Underscore to complete the template.</p> <ul style="list-style-type: none"> • <code>comment</code> : string

Name	Attributes	Description
		<p>This is the template for copying the ViewerControl comments as partial HTML and is parsed using complete template variables.</p> <ul style="list-style-type: none"> • <code>copyOverlay</code> : string <p>This is the template used for overlay when a user attempts text using a touch device. and is parsed using Underscore template variables.</p> <ul style="list-style-type: none"> • <code>hyperlinkMenu</code> : string <p>This is the template used that appears when a user creates an annotation. It contains partials using Underscore to compile.</p> <ul style="list-style-type: none"> • <code>imageStampOverlay</code> : string <p>This is the template used for picker for the image stamp redactions. It contains partials using Underscore to compile.</p> <ul style="list-style-type: none"> • <code>pageRedactionOverlay</code> : <p>This is the template used for redaction selection dialog. HTML and is parsed using complete template variables.</p> <ul style="list-style-type: none"> • <code>redactionReason</code> : string <p>This is the template used that appears when a user enters from the immediate menu HTML and is parsed using complete template variables.</p>
<p><code>uiElements</code> : Object</p>	<p><optional></p>	<p>This object contains directives to default UI elements in the viewer. alternative to removing the element from the <code>viewerTemplate.html</code> file, and features that are conditionally available.</p> <p>When the viewer removes various elements, a <code>data-pcc-removable-id</code> to the HTML element. This id will match the <code>uiElements</code> object. Setting</p>

Name	Attributes	Description
		<p> <code>true</code>, or not including the key at respective elements being visible viewer. Setting the key to <code>false</code> remove that element at runtime. </p> <p> This feature is enabled for <code>viewerContextMenuTemplate.html</code>. <code>pcc-removable-id</code> keys are all default: </p> <p> Properties </p> <ul style="list-style-type: none"> <code>annotateTab</code> : boolean <option> Default: <code>true</code> Show or hide the Annotate Tab <code>redactTab</code> : boolean <option> Default: <code>true</code> Show or hide the Redact Tab <code>searchTab</code> : boolean <option> Default: <code>true</code> Show or hide the Search Tab <code>viewTab</code> : boolean <option> Default: <code>true</code> Show or hide the View Tab <code>esignTab</code> : boolean <option> Default: <code>true</code> Show or hide the E-Sign Tab <code>copyPaste</code> : boolean <option> Default: <code>true</code> Show or hide the Text Selection <code>download</code> : boolean <option> Default: <code>true</code> Show or hide the Download Tab <code>printing</code> : boolean <option> Default: <code>true</code> Show or hide the Print Button <code>advancedSearch</code> : boolean

Name	Attributes	Description
		<p>Default: <code>false</code></p> <p>Enables the advanced search searching through marks and attachments.</p> <ul style="list-style-type: none"> • <code>attachments</code> : boolean <optional> Default: <code>true</code> <p>If <code>true</code>, the viewer will automatically load attachments of the current document.</p> <ul style="list-style-type: none"> • <code>fullScreenOnInit</code> : boolean <optional> Default: <code>false</code> <p>Specifies whether the viewer window is in full screen when initialized. If set to <code>false</code> the viewer will use the height set in <code>viewer.css</code>.</p> <ul style="list-style-type: none"> • <code>comparisonTools</code> : string <optional> Default: <code>"availableIfRevision"</code> <p>Sets the mode of the comparison tool. The following options are available:</p> <ul style="list-style-type: none"> ◦ <code>"notAvailable"</code>: No comparison tool panel is available to the user. ◦ <code>"available"</code>: Toggle comparison tool must be opened by clicking on the icon. ◦ <code>"active"</code>: Toggle comparison tool is displayed on initialization. ◦ <code>"availableIfRevision"</code>: Comparison tool is shown if revisions exist and opened by clicking on the icon. ◦ <code>"activeIfRevisions"</code>: Comparison tool is shown if revisions exist, panel is active as revisions are returned.
<code>immediateActionMenuMode</code> : string	<optional> Default: <code>"off"</code>	<p>Sets the mode of the immediate action menu. The following options are available:</p> <ul style="list-style-type: none"> • <code>"on"</code>: The menu will appear on clicking or selecting text, close to the text, allowing the user to take actions. • <code>"hover"</code>: In supported browsers, the menu will appear after creating a mark. When hovering over this icon, the menu will appear. On mobile viewports, the menu will be the same as <code>"on"</code>. • <code>"off"</code>: The menu will not appear.

Name	Attributes	Description
<code>immediateActionMenuActionsFilter</code> : object	<optional>	When <code>immediateActionMenu</code> "hover", this list will be used to determine which actions are available in the menu. Properties <ul style="list-style-type: none"> ● <code>comment</code> : boolean <optional> Default: <code>true</code> Show or hide the Add Comment Button ● <code>select</code> : boolean <optional> Default: <code>false</code> Show or hide the Select Button ● <code>copy</code> : boolean <optional> Default: <code>true</code> Show or hide the Copy Button ● <code>highlight</code> : boolean <optional> Default: <code>true</code> Show or hide the Highlight Button ● <code>redact</code> : boolean <optional> Default: <code>true</code> Show or hide the Redact Button ● <code>hyperlink</code> : boolean <optional> Default: <code>true</code> Show or hide the Hyperlink Button ● <code>cancel</code> : boolean <optional> Default: <code>false</code> Show or hide the Cancel Button
<code>commentsPanelMode</code> : string	<optional> Default: <code>"auto"</code>	Sets the mode of the comments panel. The following options are available: <ul style="list-style-type: none"> ● <code>"full"</code>: The entire content of the document is displayed in the sidebar of the document. ● <code>"skinny"</code>: An icon is placed in the sidebar of the document, representing each comment. When the icon is clicked, the comment is expanded to show the full content. ● <code>"auto"</code>: This mode will alternate between the full and skinny modes.

Name	Attributes	Description
		optimize the space available in the document.
<code>stickyTools</code> : Object	<optional> Default: "default"	<p>Set the mode for the sticky tools</p> <p>The following options are available</p> <ul style="list-style-type: none"> • "on": Sticky tools will always remain active until the user clicks on a tool. • "off": Sticky tools will always remain active for one use. • "default": Clicking on a menu item will toggle between sticky and non-sticky. The first click will trigger the tool to be on and every following click will toggle it on and off. <p>See the <code>stickyToolsFilter</code> option for more information.</p>
<code>stickyToolsFilter</code> : Object	<optional>	<p>When <code>stickyTools</code> is set to "on" or "off", this object will be used to define which tools are included in the sticky behavior. By default, all drawing tools are included. <code>EllipseAnnotation</code>, <code>RectangleAnnotation</code>, and <code>TextAnnotation</code> will have this enhanced behavior. Tools not on this list (and are not included in the <code>stickyTools</code> list or tools set to <code>false</code>, will never have this behavior.</p> <p>This object can have any of the values from <code>PCCViewer.MouseTool.Type</code>, using <code>true</code> for enabled, and <code>false</code> for disabled.</p> <p>See the <code>stickyTools</code> option for more information.</p> <p><i>Note that for some tools, like <code>Mag</code>, <code>PanAndEdit</code>, this mode is not relevant. Keeping them on or off the list will not affect their behavior.</i></p> <p>Example: <code>{ RectangleAnnotation: true, PlaceSignature: false }</code></p>
<code>signatureCategories</code> : string	<optional>	<p>Specifies the categories of eSignatures. This is a comma separated string of signature categories that the application will pass in, the UI to select categories.</p> <p>Example: "Full, Initials, Name, ..."</p>
<code>commentDateFormat</code> :	<optional>	Specifies the date format to use for comments.

Name	Attributes	Description
external:jQuery.fn~DateFormat	Default: "MM/DD/YYYY h:mma"	dates.
signatureDateFormat : external:jQuery.fn~DateFormat	<optional> Default: "MM/DD/YYYY"	Specifies the date format to use for signatures.
predefinedSearch : Object	<optional>	<p>Specifies options so that the viewer displays a document with search terms. An object is defined in the <code>predefinedSearch</code> property in all HTML5 viewer samples.</p> <p>Properties</p> <ul style="list-style-type: none"> highlightColor : string <optional> Default: <code>''</code> The default highlight color for search terms. This is overridden by the term-level <code>highlightColor</code> property. This must be in 6 digit hex format, preceded by a #. Example: <code>"#ee3a8c"</code> searchOnInit : boolean <optional> Default: <code>false</code> Whether to run the search when the viewer is initialized. Only search terms are selected: <code>true</code> will be used if the <code>search</code> property is enabled. fixed : boolean <optional> Default: <code>false</code> The default fixed value of the search panel. If <code>true</code>, overridden by the term-level <code>fixed</code> property. If <code>true</code>, the search terms are displayed and performing a search. globalOptions : Object <optional> Default: <code>{}</code> terms : Array. external:jQuery.fn~predefinedSearch <optional> Default: <code>[]</code> An array of search terms to be used for search.
searchResultsPageLength : number	<optional> Default: 250	The number of search results to show in the search results panel. The user can use the navigation buttons to page through the results. A positive number is not specified, the default is used.
revisionsPageLength : number	<optional>	The number of revisions to show in the revisions panel.

Name	Attributes	Description
	Default: 250	revisions panel. The user can use navigation buttons to page through revisions. If a positive number is not specified, the default is used.

predefinedSearchTerm

The following are options available for the Objects used in the `predefinedSearch.terms` Array. Any options not specified on the individual search term Object will use the defined property on the `predefinedSearch` Object, or a viewer default if one does not exist.

Type:

- Object

Properties:

Name	Attributes	Description
<code>searchTerm</code> : string		The term to use for search. If used along with <code>userDefinedRegex</code> , this becomes the name of the Regular Expression, while the Regular Expression is used to perform the search.
<code>highlightColor</code> : string	<optional>	The color to use when highlighting results from this particular search.
<code>fixed</code> : boolean	<optional> Default: false	Whether or not the search term is always included when performing a search. If set to true, the value of <code>predefinedSearchTerm.selected</code> is disregarded since the search term is always included.
<code>options</code> : Object	<optional>	This is the same <code>options</code> object, overriding the settings in <code>globalOptions</code> for this specific term.
<code>userDefinedRegex</code> : string	<optional>	<p>A regular expression that will be searched in place of <code>searchTerm</code>. The first and last forward slashes, as well as the flags, are stripped from the string. For example, <code>"/Pa(\w+)/ig"</code> will become <code>"Pa(\w+)"</code>.</p> <p>When special characters (ex: backslash) are used in the "userDefinedRegex" field, they need to be properly escaped. For example, for searching words that begins with "Pa", the regular expression will be <code>"Pa(\w+)"</code>, this regular expression should be properly escaped like this <code>"Pa(\w+)"</code>.</p> <p>All patterns use the Global (g) flag.</p> <p>Lines of text in the Viewer typically end with a space followed by a <code>\n</code> character, so to include phrases that span multiple lines in your regular expression search results, you will need to provide a regular expression that accounts for words separated by either a space or a space followed by a <code>\n</code> character.</p>

Name	Attributes	Description
<code>selected</code> : boolean	<optional> Default: false	Whether or not this term is selected in the Patterns menu by default. This property must be set to true if you expect to use this search terms along with <code>searchOnInit: true</code> .

redactionReasons

An object defining the redaction reasons to use in the viewer. See the "Using Redaction Reasons" section of this help file for more information.

Type:

- Object

Properties:

Name	Attributes	Description
<code>enableRedactionReasonSelection</code> : boolean	<optional> Default: true	Determines whether the UI for selecting redaction reasons is enabled. When <code>false</code> , the UI for selecting redaction reasons will not be available.
<code>reasons</code> : Array.<Object>		<p>Array of objects defining the redaction reasons which the user may select from. Each item may contain:</p> <p>Properties</p> <ul style="list-style-type: none"> • <code>reason</code> : string Reason text to apply to the redaction. • <code>description</code> : string Reason description to show in the reason selection UI. • <code>defaultReason</code> : boolean <optional> Default: <code>false</code> Determines whether this redaction should be selected by default. When <code>enableMultipleRedactionReasons</code> is <code>true</code>, can be set to <code>true</code> for multiple items in the <code>reasons</code> array. When <code>enableMultipleRedactionReasons</code> is <code>false</code>, can only be set to <code>true</code> for one item in the <code>reasons</code> array.
<code>autoApplyDefaultReason</code> : boolean	<optional>	When <code>true</code> , default reason(s) will automatically

Name	Attributes	Description
	Default: false	be applied to new redactions.
<code>enableFreeformRedactionReasons</code> : boolean	<optional> Default: false	When <code>true</code> , users are given the option to type a freeform redaction reason.
<code>maxLengthFreeformRedactionReasons</code> : number	<optional> Default: false	Maximum length of a typed redaction reason.
<code>enableMultipleRedactionReasons</code> : boolean	<optional> Default: false	Determines whether multiple redaction reasons can be applied to redactions. When <code>false</code> , only a single redaction reason can be applied to a redaction. When <code>true</code> , multiple redaction reasons can be applied to a redaction (requires <code>annotationsMode</code> be set to "LayeredAnnotations").

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:09 GMT-0400 (Eastern Daylight Time)

Namespace: PCCViewer

PCCViewer

`PCCViewer` is the global namespace used for members of this API.

Classes

- [AjaxResponse](#)
- [BurnRequest](#)
- [Comment](#)
- [Conversation](#)
- [ConversionRequest](#)
- [DocumentHyperlink](#)
- [Error](#)
- [Event](#)
- [ImageStamps](#)
- [LoadMarkupLayersRequest](#)
- [Mark](#)
- [MarkupLayer](#)
- [MarkupLayerCollection](#)
- [MouseTool](#)
- [ObservableCollection](#)

PrintRequest
Promise
Revision
RevisionsRequest
SearchRequest
SearchResult
SearchTask
SearchTaskResult
SignatureControl
SignatureDisplay
ThumbnailControl
Viewer
ViewerControl

Mixins

Data
SessionData

Namespaces

Ajax
Language
MouseTools
Signatures
Util

Members

(static, readonly) **EventType** :string

The `EventType` enumeration defines event types known to [PCCViewer.ViewerControl](#).

Note: This enumeration is for convenience for API developers. Instead of using this enumeration, you can pass string values of the `eventType` (enumeration values)

Type:

- string

Properties:

Name	Description
ViewerReady : string	<p>Triggered when the Viewer is ready.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • None
PageCountReady : string	<p>Event is triggered when the viewer has an actual page count from the server and the consumer can begin to interact with the viewer interfaces.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>pageCount {number}</code> The actual page count of the document.
EstimatedPageCountReady : string	<p>Event is triggered when the viewer has an estimated page count from the server.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>pageCount {number}</code> The estimated page count of the document.
PageChanged : string	<p>Event is triggered when the viewer changed the current page.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • none
PageLoadFailed : string	<p>Event is triggered when the viewer changed the current page.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>pageNumber {number}</code> Indicates the page number of the page that failed to load. • <code>statusCode {number}</code> Indicates the HTTP page load failure error code returned by the image service • <code>accusoftErrorNumber {number}</code> The error codes in this category currently are: <ul style="list-style-type: none"> ◦ 4001 Document requires a password (HTTP statusCode will be 480) ◦ 5001 Unable to generate Page (HTTP statusCode will be 580) ◦ 5002 Download of the file to the Image service failed (HTTP statusCode will be 580) • <code>accusoftErrorMessage {string}</code> Description of the error provided by the Image service.

Name	Description
PageDisplayed : string	<p>Event is triggered when the viewer has displayed a page. If the content of a page is large, for example an engineering drawing with several hundred nodes, then the browser may be busy still rendering/preparing the content when this event gets fired. Note that if the <code>maxOutOfViewDisplay</code> viewer parameter is greater than 0, then out-of-view pages will be displayed (the page content will be loaded in the DOM, though the page will not be visible since it is out of view). In this scenario, the PageDisplayed event will fire for out-of-view pages.</p> <p>Augmented Properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>pageNumber</code> {number} The page number of the displayed page.
PageRotated : string	<p>Event is triggered when the viewer has displayed a page, not necessarily the content of a page.</p> <p>Augmented properties of the PCCViewer.ViewerControl.Event object for this event:</p> <ul style="list-style-type: none"> • <code>pageNumber</code> {number} The page number of the page that was rotated.
DocumentRotated : string	<p>Event is triggered when the rotation of all pages in the document changes.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>degreesClockwise</code> {number} The amount in degrees clockwise the pages were rotated.
ScaleChanged : string	<p>Event is triggered when the scaling of page(s) in the viewer changed. After the user actions, zoom buttons pressed, zoom api called, fit type changed, viewer mode changed and that resulted in a scale change.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>scaleType</code> {string} Gives an indication of whether the content was <code>scaledUp</code> (got bigger) or <code>scaledDown</code> (got smaller). • <code>scaleFactor</code> {number} Indicates the new scale factor of the viewer. A value of 1 indicates 100% zoom. See also PCCViewer.ViewerControl#getScaleFactor. • <code>trigger</code> {string} Indicates how the scale change was triggered. Possible values are: <ul style="list-style-type: none"> ◦ "Zoom" - Indicates a direct zoom, such as using the

Name	Description
	<p>PCCViewer.ViewerControl#zoomIn, PCCViewer.ViewerControl#zoomOut, or PCCViewer.ViewerControl#setScaleFactor methods.</p> <ul style="list-style-type: none"> o "Fit" - Indicates a change due to a fit type being applied through the PCCViewer.ViewerControl#fitContent method. • <code>fitType</code> {string} Indicates the fit type that was applied, if applicable. This property will only be defined if the trigger was Fit, and will be undefined otherwise.
<code>DocumentPrinted</code> : string	<p>Event is triggered when the print button was clicked in the viewer's print dialog. We have no way to know if the page printed in the system print dialog.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>pageNumbers</code> {Array.<number>} An array containing the page number of each page that was printed. NOTE: In the PageView viewer, the array contains the current page only. • <code>orientation</code> {string} "portrait" or "landscape" • <code>includeMarks</code> {boolean} Indicates whether the marks were included in the printed pages.
<code>TextSelected</code> : string	<p>Event is triggered when text is selected.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>selectedText</code> {string} Deprecated since v9.2 (use the <code>textSelection.text</code> argument instead). • <code>pageNumbers</code> {Array.<number>} Deprecated since v9.2 (use the <code>textSelection.pageNumber</code> argument instead). • <code>textSelection</code> {PCCViewer.ViewerControl.TextSelection} An object that provides information regarding the text selection. • <code>clientX</code> {number} An optional value indicating the absolute window position in the x-axis of the cursor at the end of the selection. A value for this property is available only when using the <code>SelectText</code> mouse tool. • <code>clientY</code> {number} An optional value indicating the absolute window position in the y-axis of the cursor at the end of the selection. A value for this property is available only when using the <code>SelectText</code> mouse tool. • <code>handleClientX</code> {number} An optional value indicating the absolute window position in the x-axis of the handle at the sliding end of the selection. A value for this property is available when the text is initially selected or when the selection is edited. (In either case, one end of the selection

Name	Description
	<p>is stationary throughout the drag, and the other end is sliding.)</p> <ul style="list-style-type: none"> • <code>handleClientY</code> {number} An optional value indicating the absolute window position in the y-axis of the handle at the sliding end of the selection. A value for this property is available when the text is initially selected or when the selection is edited. (In either case, one end of the selection is stationary throughout the drag, and the other end is sliding.)
MouseToolChanged : string	<p>Event is triggered when the mouse tool changed. This change could be initiated through the viewer's toolbar, viewer's context menu, or viewer's API.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>mouseToolName</code> {string} Indicates the name of the new mouse tool.
SearchPerformed : string	<p>Triggered when a search is performed with a call to PCCViewer.ViewerControl#search.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>searchRequest</code> {PCCViewer.SearchRequest} The search request returned from the call to PCCViewer.ViewerControl#search.
PartialSearchResultsAvailable : string	<p>Event is triggered when partial search results are available</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>partialSearchResults</code> {Array. <PCCViewer.SearchResult>} The new search results found since the last "PartialSearchResultsAvailable" event. • <code>pagesWithoutText</code> {Array.<number>} The set of pages that could not be searched because searchable text was not available for the page. This includes only the pages on which searching was attempted since the last "PartialSearchResultsAvailable" event.
SearchCompleted : string	<p>Event is triggered when search is completed successfully, user cancelled, or an exception. This event will also return the <code>searchResult</code> object to the consumer</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>completedSearchResults</code> {Array.

Name	Description
	<PCCViewer.SearchResult>} The set of search results.
SearchFailed : string	Event is triggered when search failed to due to an exception. Augmented properties of the PCCViewer.Event object for this event: <ul style="list-style-type: none"> • errorMessage {string} A human readable error message indicating why the search failed.
SearchCancelled : string	Event is triggered when search is cancelled by the user. Augmented properties of the PCCViewer.Event object for this event: <ul style="list-style-type: none"> • none
SearchResultsAvailable : string	Event is triggered when search is completed and the results are available. This event will return the full results object to the consumer if available. Augmented properties of the PCCViewer.Event object for this event: <ul style="list-style-type: none"> • completedSearchResults {Array. <PCCViewer.SearchResult>} The set of search results.
SearchCleared : string	Event is triggered when the current search is cleared. After this event, calls to PCCViewer#setSelectedSearchResult , PCCViewer#getSelectedSearchResult , and PCCViewer#getSearchRequest will no longer be valid. Augmented properties of the PCCViewer.Event object for this event: <ul style="list-style-type: none"> • none
SearchResultSelectionChanged : string	Event is triggered when the selected search result changes, including when the first result is selected, the selection is cleared, or the selection changes from one result to another. Augmented properties of the PCCViewer.Event object for this event: <ul style="list-style-type: none"> • none
RevisionsRetrievalCompleted : string	Event is triggered when the revisions retrieval has completed due to a failure or when the full set of revisions is available. Augmented properties of the PCCViewer.Event object for this event: <ul style="list-style-type: none"> • completedRevisions {Array.<PCCViewer.Revision>} The set of all revisions.

Name	Description
RevisionsRetrievalFailed : string	<p>Event is triggered when the revisions retrieval has completed due to a failure.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>errorMessage</code> {string} A human readable error message indicating why the revisions request failed.
RevisionsAvailable : string	<p>Event is triggered when the revisions retrieval has completed because the full set of revisions is available.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>completedRevisions</code> {Array.<PCCViewer.Revision>} The set of all revisions.
PartialRevisionsAvailable : string	<p>Event is triggered when a partial set of revisions is available.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>partialRevisions</code> {Array.<PCCViewer.Revision>} The new revisions found since the last "PartialRevisionsAvailable" event
PrintRequested : string	<p>Event is triggered when a document print is requested through PCCViewer.ViewerControl#print.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>printRequest</code> {PCCViewer.PrintRequest}
MarkupLoaded : string	<p>Event is triggered when markup is loaded from a file through PCCViewer.ViewerControl#loadMarkup. Triggered when annotations are loaded from a file.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>name</code> {string} The name of the markup data that was loaded. • <code>loadedMarks</code> {Array.<PCCViewer.Mark>} The marks that were loaded.
MarkupSaved : string	<p>Event is triggered when annotations save to the server.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>name</code> {string} The name of the markup data that was saved.

Name	Description
MarkChanged : string	<p>Event is triggered when one or more attributes changes on an annotation.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>mark</code> {PCCViewer.Mark} The changed annotation object • <code>pageNumber</code> {number} The page number of the annotation. • <code>propertyNames</code> {Array.<string>} The names of properties that have changed.
MarkCreated : string	<p>Event is triggered when a new annotation is created.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>mark</code> {PCCViewer.Mark} The annotation object. • <code>pageNumber</code> {number} The page number of the annotation. • <code>clientX</code> {number} An optional value indicating the absolute window position in the x-axis of the cursor at the end of the selection. A value for this property is available only when using a mouse tool to create the mark. Values will be <code>undefined</code> for marks added using the API. • <code>clientY</code> {number} An optional value indicating the absolute window position in the y-axis of the cursor at the end of the selection. A value for this property is available only when using a mouse tool to create the mark. Values will be <code>undefined</code> for marks added using the API.
MarkRemoved : string	<p>Event is triggered when a annotation is removed from a page.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>mark</code> {PCCViewer.Mark} The annotation object. • <code>pageNumber</code> {number} The page number of the annotation.
MarkReordered : string	<p>Event is triggered when the annotation's stacking order has changed.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>mark</code> {PCCViewer.Mark} The annotation object. • <code>pageNumber</code> {number} The page number of the annotation. • <code>index</code> {number} The new stacking order index of the annotation. • <code>oldIndex</code> {number} The old stacking order index of the annotation.

Name	Description
<code>MarkSelectionChanged</code> : string	<p>Triggered when the set of selected annotations has changed.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>pageNumber</code> {number} The page number containing the mark that was selected or deselected.
<code>MarkMouseEnter</code> : string	<p>Event is triggered when the mouse enters the annotation bounding box.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>mark</code> {PCCViewer.Mark} The annotation object. • <code>clientX</code> {number} A value indicating the absolute window position in the x-axis of the cursor. • <code>clientY</code> {number} A value indicating the absolute window position in the y-axis of the cursor.
<code>MarkMouseOver</code> : string	<p>Event is triggered when the mouse moves over the annotation bounding box.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>mark</code> {PCCViewer.Mark} The annotation object. • <code>clientX</code> {number} A value indicating the absolute window position in the x-axis of the cursor. • <code>clientY</code> {number} A value indicating the absolute window position in the y-axis of the cursor.
<code>MarkMouseLeave</code> : string	<p>Event is triggered when the mouse leaves the annotation bounding box.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>mark</code> {PCCViewer.Mark} The annotation object. • <code>clientX</code> {number} A value indicating the absolute window position in the x-axis of the cursor. • <code>clientY</code> {number} A value indicating the absolute window position in the y-axis of the cursor.
<code>CommentsPanelToggled</code> : string	<p>Triggered when the comments panel opens or closes.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>isOpen</code> {boolean} Whether the comments panel is open or closed.

Name	Description
<code>CommentCreated</code> : string	<p>Triggered when a comment is added to a Conversation in the viewer.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>conversation</code> {PCCViewer.Conversation} The Conversation containing the changed comment. • <code>comment</code> {PCCViewer.Comment} The comment that was added.
<code>CommentRemoved</code> : string	<p>Triggered when a comment is removed from a Conversation in the viewer.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>conversation</code> {PCCViewer.Conversation} The Conversation containing the changed comment. • <code>comment</code> {PCCViewer.Comment} The comment that was removed.
<code>CommentChanged</code> : string	<p>Triggered when the text of a comment in the viewer has changed.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>conversation</code> {PCCViewer.Conversation} The Conversation containing the changed comment. • <code>comment</code> {PCCViewer.Comment} The comment that was modified.
<code>PageTextReady</code> : string	<p>Triggered when the text of a page has been loaded in the viewer.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>pageNumber</code> {number} The page number of the page that text is ready for.
<code>Click</code> : string	<p>Triggered when a user clicks a page or comment pane in the viewer.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>pageNumber</code> {number} The page number of the page that was clicked, or null if none. • <code>targetType</code> {string} A description of the clicked object: "mark", "searchResult", "textSelection", "page", "comments", "documentHyperlink" or null (if the user clicked in an area outside of a page and outside of the comments panel).

Name	Description
	<ul style="list-style-type: none"> • <code>textSelection</code> {object} The text selection that was clicked, or null if none. • <code>mark</code> {object} - The mark object that was clicked, or null if none. • <code>searchResult</code> {object} - The search result that was clicked, or null if none. • <code>documentHyperlink</code> {PCCViewer.DocumentHyperlink} - The document hyperlinks that was clicked, or null if none. • <code>originalEvent</code> {object} - A copy of the browser event. • <code>clientX</code> {number} - The x window coordinate of the position where the user clicked. • <code>clientY</code> {number} - The y window coordinate of the position where the user clicked.
PageOpening : string	<p>Triggered when the width and height page attributes are retrieved. Note that this event will fire whenever a page opens, so if a page opens, it will fire, and if the page is scrolled out of view, disposed, and then scrolled back into view, the event will fire again.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • <code>width</code> {number} The width in pixels of the page that is opening • <code>height</code> {number} The height in pixels of the page that is opening • <code>pageNumber</code> {number} The 1-based number of the page that is opening
ViewingSessionChanging : string	<p>Fires when the ViewerControl begins to change to a new viewing session but before the new viewing session is ready. During this time, most ViewerControl API calls will fail with an error.</p> <p>A subsequent <code>ViewingSessionChanged</code> event will fire indicating that the change to the new viewing session has completed and the ViewerControl API is ready to be used again.</p> <p>Note: The viewing session is typically changed to present a different document to the end user (for example, when the navigating into an email attachment). The ViewerControl can be instructed to change to a new viewing session by a call to PCCViewer.ViewerControl#changeViewingSession. Calling this method will immediately fire this event, indicating that the current viewing session is about to change.</p> <p>Augmented properties of the <code>PCCViewer.Event</code> object for this event:</p> <ul style="list-style-type: none"> • <code>viewingSessionId</code> {string} Id of the viewing session being switched to.

Name	Description
ViewingSessionChanged : string	<p>Fires when the ViewerControl has finished changing to a new viewing session and the ViewerControl API is ready to be used again (see the related <code>ViewingSessionChanging</code> event).</p> <p>A subsequent <code>PageCountReady</code> event will fire when the new page count is ready.</p> <p>Augmented properties of the <code>PCCViewer.Event</code> object for this event:</p> <ul style="list-style-type: none"> <code>viewingSessionId</code> {string} Id of the new viewing session in use.

See: [PCCViewer.ViewerControl#on](#)
[PCCViewer.ViewerControl#off](#)

(static, readonly) **FitType** :string

The `FitType` enumeration defines fit types known by `PCCViewer.ViewerControl`. The `ViewerControl` uses a specified fit type to set or update the scaling of the pages displayed in the viewer.

Note: This enumeration is a convenience for API developers. Instead of using it, you can pass in the string values of the fit type (enumeration values) directly to the API.

Type:

- string

Properties:

Name	Description
FullWidth : string	The viewer scales the content to fill the width of the viewer.
ShrinkToWidth : string	The viewer will scale down the content until it fits fully width-wise into view. The page will not be scaled up if it already fits.
ActualSize : string	The viewer shows the content actual size. The content is not scaled.
FullHeight : string	The viewer scales the content to fill the height of the viewer, based on the largest known page height.
FullPage : string	The viewer scales the content to best fit the largest known page in the viewer.

See: [PCCViewer.ViewerControl#fitContent](#)

Example

```
// use the enumeration
myViewerControl.fitContent(PCCViewer.FitType.FullWidth);

// or just use the string value
myViewerControl.fitContent("FullWidth");
```

(static, readonly) **MarkHandleMode** :string

The MarkHandleMode enumeration defines mark handle modes known by [PCCViewer.ViewerControl](#). The ViewerControl uses a specified mark handle mode to determine how the mark handles are shown.

Note: This enumeration is a convenience for API developers. Instead of using it, you can pass in the string values of the mode (enumeration values) directly to the API.

Type:

- string

Properties:

Name	Description
HideSideHandlesWhenClose : string	All 8 handles (top left, top, top right, left, right, bottom left, bottom, bottom right) are shown for rectangular marks, except when the handles are moved close to each other. If the left handles are close to the right handles, the top and bottom handles are not shown. If the top handles are close to the bottom handles, the left and right handles are not shown.
HideCornerHandlesWhenClose : string	All 8 handles (top left, top, top right, left, right, bottom left, bottom, bottom right) are shown for rectangular marks, except when the handles are moved close to each other. In that case, the corner handles (top left, top right, bottom left, bottom right) are not shown.

See: [PCCViewer.ViewerControl#getMarkHandleMode](#)
[PCCViewer.ViewerControl#setMarkHandleMode](#)

Example

```
// use the enumeration
myViewerControl.setMarkHandleMode(PCCViewer.MarkHandleMode.Hid

// or just use the string value
myViewerControl.setMarkHandleMode("HideCornerHandlesWhenClose")
```

(static, readonly) **PageLayout** :string

The PageLayout enumeration defines page layouts known by [PCCViewer.ViewerControl](#). The

`ViewerControl` uses a specified page layout to set or update the placement or arrangement of the pages in the viewer.

Note: This enumeration is a convenience for API developers. Instead of using it, you can pass in the string values of the page layout (enumeration values) directly to the API.

Type:

- string

Properties:

Name	Description
Horizontal : string	Pages are displayed as a single horizontal row and a horizontal scroll bar is displayed to bring into view the pages that are not in view.
Vertical : string	Pages are displayed as a single vertical column and a vertical scroll bar is displayed to bring into view the pages that are not in view.

See: `PCCViewer.ViewerControl#pageLayout`

Example

```
// use the enumeration
myViewerControl.setPageLayout(PCCViewer.PageLayout.Horizontal);

// or just use the string value
myViewerControl.setPageLayout("Horizontal");
```

(static, readonly) `RedactionViewMode` :string

The `RedactionViewMode` enumeration defines redaction view modes known by `PCCViewer.ViewerControl`. The `ViewerControl` uses a specified redaction view mode to set visibility of the text underneath the redaction rectangle marks that are opaque.

Note: This enumeration is a convenience for API developers. Instead of using it, you can pass in the string values of the view mode (enumeration values) directly to the API.

Type:

- string

Properties:

Name	Description
Draft : string	The viewer displays the document content underneath the redaction rectangles in the document.
Normal :	The viewer hides the document content underneath the redaction rectangles.

Name	Description
string	

See: [PCCViewer.ViewerControl#getRedactionViewMode](#)
[PCCViewer.ViewerControl#setRedactionViewMode](#)

Example

```
// use the enumeration
myViewerControl.setRedactionViewMode(PCCViewer.ViewMode.Draft);

// or just use the string value
myViewerControl.setRedactionViewMode("Draft");
```

(static, readonly) **ScaleTrigger** :string

The `ScaleTrigger` enumeration defines actions known to `PCCViewer.ViewerControl` that alter page scaling.

Note: This enumeration is for convenience for API developers. Instead of using this enumeration, you can pass string values of the `eventType` (enumeration values)

Type:

- string

Properties:

Name	Description
Pinch : string	
Zoom : string	
Fit : string	
ViewMode : string	

Example

```
// use the enumeration
ev.trigger === PCCViewer.ScaleTrigger.Pinch

// or just use the string value
ev.trigger === "Pinch"
```

(static, readonly) **ViewMode** :string

The ViewMode enumeration defines view modes known by [PCCViewer.ViewerControl](#). The `ViewerControl` uses a specified view mode to set or update how documents that contain different sized pages are displayed in the viewer.

Note: This enumeration is a convenience for API developers. Instead of using it, you can pass in the string values of the view mode (enumeration values) directly to the API.

Type:

- string

Properties:

Name	Description
Document : string	The viewer maintains the relative size of each page when displaying a document. For example, if page 2 is smaller than page 1, it will appear smaller.
EqualWidthPages : string	Deprecated since v10.0 (use the "EqualFitPages" enumeration value instead).
SinglePage : string	The viewer displays a single page at a time. Each page is scaled to fit within a view box, which is the initial size of the viewer and increases in size when zooming in (and decreases in size when zooming out). After the viewer initializes, the view mode may not be changed to or from SinglePage view mode (an Error will be thrown in this case).
EqualFitPages : string	The viewer scales each page so that their width is the same, when using vertical page layout. For example, if page 2 is smaller than page 1, it will be scaled larger so that its width is equal to the width of page 1. If using horizontal page layout, the viewer scales each page so that their height is the same.

See: [PCCViewer.ViewerControl#getViewMode](#)
[PCCViewer.ViewerControl#setViewMode](#)

Example

```
// use the enumeration
myViewerControl.setViewMode(PCCViewer.ViewMode.EqualFitPages);

// or just use the string value
myViewerControl.setViewMode("EqualFitPages");
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: AjaxResponse

PCCViewer. AjaxResponse

`new AjaxResponse(response)`

This object provides a public API for AJAX Responses. It should be instantiated with every new AJAX response inside of Viewer Control.

Parameters:

Name	Type	Description															
response	Object	Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>headers</td><td>Object</td><td></td></tr><tr><td>status</td><td>Number</td><td></td></tr><tr><td>statusText</td><td>String</td><td></td></tr><tr><td>responseText</td><td>String</td><td></td></tr></tbody></table>	Name	Type	Description	headers	Object		status	Number		statusText	String		responseText	String	
Name	Type	Description															
headers	Object																
status	Number																
statusText	String																
responseText	String																

Properties:

Name	Description
<code>status</code> : Number	
<code>statusText</code> : String	
<code>responseText</code> : String	

Methods

`getResponseHeader(header) → {String}`

Gets the value of a header

Parameters:

Name	Type	Description
header	String	

Returns:

Type
String

Example

```
// Get the header with a case-insensitive argument
var response = new PCCViewer.AjaxResponse({
  headers: { 'Content-Type': 'application/json', 'X-
Powered-By': 'Express' },
  status: 200,
  statusText: 'OK',
  responseText: 'Success!'
});

response.getResponseHeader('Content-Type'); // returns
'application/json'
response.getResponseHeader('content-type'); // returns
'application/json'
response.getResponseHeader('CONTENT-TYPE'); // returns
'application/json'
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: BurnRequest

PCCViewer. BurnRequest

(protected) **new BurnRequest()**

The `BurnRequest` object is created when burning redactions/signatures in a document.

The `BurnRequest` is a thenable object, which allows Promise-like interactions. Calling the [PCCViewer.BurnRequest#then](#) method will return a [PCCViewer.Promise](#) object. On successful burn, the `Promise` success method, if added, is called with `url` to download the burned document. On a burn failure, the `Promise` is rejected.

The `BurnRequest` object also provides an event subscription method, to get notified of other types of information. See [PCCViewer.BurnRequest.EventType](#).

Note: This constructor should not be used directly. Instead, a burn request is created by a call to [PCCViewer.ViewerControl#burnMarkup](#).

Example

```
function onSuccessfullBurn(burnturl) {
  alert("burntURL = " + burnturl);
  console.log(burnturl);
}

function onFailedBurn(error) {
```

```

    alert("burn Process failed, reason:" + (error.message ?
error.message : error));
}

// A BurnRequest object is created by and returned from the
call to the burnMarkup method
var burnRequest = viewerControl.burnMarkup();
burnRequest.then(onSuccessfulBurn, onFailedBurn);

//register some events
burnRequest
    .on(PCCViewer.BurnRequest.EventType.BurnCompleted,
        function(ev) {
            alert("Document burn completed.");
        })
    .on(PCCViewer.BurnRequest.EventType.BurnProgress,
        function(event) {
            alert("Burn progress: " + event.percent + "%");
        })
    .on(PCCViewer.BurnRequest.EventType.BurnFailed,
        function(event) {
            alert("Document burn failed.");
        });

```

Members

(static) **EventType** :string

A list of events that can be triggered by the [PCCViewer.BurnRequest](#) object.

Type:

- string

Properties:

Name	Description
BurnProgress : string	Triggered when the burn process receives an update. Augmented properties of the PCCViewer.Event object for this event: <ul style="list-style-type: none"> • percent {number} - Indicates the estimated percentage complete.
BurnCompleted : string	Triggered when the burn process completes successfully, fails to complete, or is cancelled. Augmented properties of the PCCViewer.Event object for this event: None
BurnFailed : string	Triggered when the burn process fails. Augmented properties of the PCCViewer.Event object for this event: None
BurnCancelled :	Triggered when user cancels burn process.

Name	Description
string	Augmented properties of the PCCViewer.Event object for this event: None
BurnWorkerCreated : string	Triggered when the burn process is created by the backend burn service. Augmented properties of the PCCViewer.Event object for this event: None

(readonly) **burnedDocumentDownloadURL** :string|null

Gets the URL for downloading the burned document after successful burn.

This property is defined on all BurnRequest objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string | null

See: [PCCViewer.BurnRequest#getBurnedDocumentDownloadURL](#)

Example

```
var downloadUrl = burnRequest.burnedDocumentDownloadURL;
```

(readonly) **errorCode** :number

Gets the errorCode if there is a failure during burn process.

This property is defined on all BurnRequest objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.BurnRequest#getErrorCode](#)

Example

```
var errorCode = burnRequest.errorCode;
```

(readonly) **options** :Object

Gets the options that were provided/used to burn the document.

This property is defined on all BurnRequest objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.BurnRequest#getOptions](#)

Example

```
var options = burnRequest.options;
```

(readonly) **progress** :number

Gets the current estimate of the burn process progress.

This property is defined on all BurnRequest objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.BurnRequest#getProgress](#)

Example

```
var percentProgress = burnRequest.progress;
```

(readonly) **workerID** :number

Gets the worker ID.

This property is defined on all BurnRequest objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.BurnRequest#getWorkerID](#)

Example

```
var percentProgress = burnRequest.workerID;
```

Methods**cancel()** → {[PCCViewer.BurnRequest](#)}

Cancels the burn request. The `PCCViewer.Promise` object that is returned will be rejected. If the user cancels the operation, then a `PCCViewer.Error` object will be returned as the rejection reason and its `code` property will be set to `UserCancelled`.

Returns:

The cancelled request.

Type

[PCCViewer.BurnRequest](#)

Example

```
var burnRequest = viewerControl.burnMarkup();
burnRequest.cancel();
```

getBurnedDocumentDownloadURL() → {[string|null](#)}

Gets a URL for a web tier service to download the burned document.

Returns:

Returns a URL to download the burned document. If called before the process completes, or after the process fails, it will return `null`.

Type

`string | null`

Example

```
var burnRequest = viewerControl.burnMarkup();
var returnedUrl =
burnRequest.getBurnedDocumentDownloadURL();
```

getErrorCode() → {[string](#)}

Gets the error code for the burn request, in cases where the request has failed.

Returns:

A value for programmatic identification of an error condition, or null if an error has not occurred. If the error was in the underlying PCCIS MarkupBurner, the error code is the PCCIS error code:

- DocumentFileIdError
- DocumentFileIdDoesNotExist
- MarkupFileIdError
- MarkupFileIdDoesNotExist
- User Cancelled
- Failure to generate Markup XML

Type

string

Example

```
var burnRequest = viewerControl.burnMarkup();
var errorCode = burnRequest.getErrorCode();
```

getOptions() → {Object}

Gets a copy of the options object used by the burn request.

The original options object has been provided to the method [PCCViewer.ViewerControl#burnMarkup](#). If no options object was provided to `burnMarkup`, or the options object did not define all properties, then the returned object will represent the actual options used.

Returns:

A copy of the burn options object which is used by this burn request.

- `burnSignatures {boolean}` - Whether to burn the signatures.
- `burnRedactions {boolean}` - Whether to burn the redactions.
- `filename {string}` - (optional) Filename for the burned document.
- `removeFormFields {Array.}` - (optional) List of types of form fields to remove.

Type

Object

Example

```
var burnRequest = viewerControl.burnMarkup();
var options = burnRequest.getOptions();
```


`getProgress()` → {number}

Gets the currently known burn progress value.

Returns:

A number between 0 and 100 (inclusive). A value of 100 means the burn process completed.

Note: In the 9.1 release, the values provided are 0 or 100. The value 0 indicates that the burn process is incomplete.

Type
number

Example

```
var burnRequest = viewerControl.burnMarkup();
var percent = burnRequest.getProgress();
```

`getWorkerID()` → {string|null}

Gets the ID of the PCCIS MarkupBurner performing burning.

Returns:

The ID of the PCCIS MarkupBurner performing burning. Returns `null` before the worker is created.

Type
string | null

Example

```
var burnRequest = viewerControl.burnMarkup();
var workerId = burnRequest.getWorkerID();
```

`off()` → {PCCViewer.BurnRequest}

Remove event listeners from the `BurnRequest` object.

See: [PCCViewer.ViewerControl#off](#) for more on how it is used.
[PCCViewer.BurnRequest.EventType](#) for a list of events.

Returns:

The object on which this method was called.

Type

[PCCViewer.BurnRequest](#)**on()** → **{[PCCViewer.BurnRequest](#)}**Add event listeners to the `BurnRequest` object.

See: [PCCViewer.BurnRequest.EventType](#) for a list of events.
[PCCViewer.ViewerControl#on](#) for more detailed examples.

Returns:

The object on which this method was called.

Type

[PCCViewer.BurnRequest](#)**Example**

```
var burnRequest = viewerControl.burnMarkup();
burnRequest
  .on(PCCViewer.BurnRequest.EventType.BurnCompleted,
    function(ev) {
      alert("Document burn completed.");
    })
  .on(PCCViewer.BurnRequest.EventType.BurnProgress,
    function(event) {
      alert("Burn progress: event.percent + \"%");
    });
```

then([onFulfilled_{opt}](#), [onRejected_{opt}](#)) → **{[PCCViewer.Promise](#)}**Register callbacks to access the current or eventual result of the `BurnRequest`.On successful burn, the `onFulfilled` callback(s) are called with the URL to download the burned document.On a burn failure, the `onRejected` callback(s) are called with a [PCCViewer.Error](#).Multiple `onFulfilled` or `onRejected` callbacks can be registered for the same `BurnRequest` by calling this method multiple times.**Parameters:**

Name	Type	Attributes	Description
<code>onFulfilled</code>	PCCViewer.Promise~onFulfilled	<optional>	Called if or when the promise is resolved. Optionally pass a value of <code>null</code> or <code>undefined</code> if you do not use this callback, but you want to provide an <code>onRejected</code> callback.

Name	Type	Attributes	Description
onRejected	PCCViewer.Promise~onRejected	<optional>	Called if or when the promise is rejected.

Returns:

A promise object that is resolved according to the Promises/A+ standard.

Type

[PCCViewer.Promise](#)

Example

```
var viewerControl =
$("#myElement").pccViewer(...).viewerControl;

// a basic example
viewerControl.burnMarkup().then(
  function onFulfilled(url) {
    // download document using `url`
  },
  function onRejected(error) {
    // Handle failure
  }
);
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: Comment

PCCViewer. Comment

new Comment(comment, conversation)

The constructor for a `Comment` Object. This describes the comments that belong to a mark [PCCViewer.Conversation](#).

It will not be necessary to create comments directly with this constructor. Rather, simply use [PCCViewer.Conversation#addComment](#).

Note: Comment text is not sanitized in any way by the API, and will exist as the same string value assigned to it. To ensure security of the web application, text data may need to be sanitized and safely inserted into the DOM when it is used.

Parameters:

Name	Type	Description
comment	string	The text content of the comment.
conversation	PCCViewer.Conversation	The PCCViewer.Conversation Object that this comment belongs to.

Throws:

- If `commentText` is not a string.

Type
Error

- If `conversation` is not a [PCCViewer.Conversation](#) Object.

Type
Error

Example

```
// assume we already have a PCCViewer.Mark object created
var conversation = mark.getConversation();

var comment = conversation.addComment('This is the best
comment ever.');
```

Members

creationTime :string

Gets and sets the date and time when the comment was created.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Comment#getCreationTime](#)
[PCCViewer.Comment#setCreationTime](#)

text :string

Gets and sets the text content of the Comment.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Comment#getText](#)
[PCCViewer.Comment#setText](#)

Methods

[getConversation\(\)](#) → [{PCCViewer.Conversation}](#)

Gets the conversation that this comment is (or was) a part of.

Returns:

The Comment's Conversation.

If a comment is deleted from a conversation, this still returns the Conversation object.

Type

[PCCViewer.Conversation](#)

[getCreationTime\(\)](#) → [{Date}](#)

Gets the date and time when the comment was created.

See: [PCCViewer.Comment#setCreationTime](#)

Returns:

A JavaScript Date Object representing the creation time of the comment.

Type

Date

[getData\(key\)](#) → [{string|object}](#)

Gets the data value for the given key, or gets a hash containing all key values, if a key was not provided.

Note: If a hash is returned, this will be a new object each time it is called. Adding new properties to the returned hash will not add data to the Comment.

Note: The returned data is not mutated or sanitized, which could lead to a security vulnerability if not sanitized properly before use.

Parameters:

Name	Type	Description
key	string	The key for which to get the data value.

See: [PCCViewer.Data#getData](#)
[PCCViewer.Comment#setData](#)
[PCCViewer.Comment#getDataKeys](#)

Throws:

If the `key` argument is null or otherwise not a string.

Type
Error

Returns:

- If a key argument was provided, it returns the associated value.
- If a key argument was provided, but a value has not been associated with the key, then it returns undefined.
- If a key was not provided, it returns a hash object containing all key-value pairs.

Type
string | object

Example

```
var comment = myConversation.addComment("Hello world.");

// The key "Author" is set the value "Mark".
comment.setData("Author", "Mark");

// The key "Severity" is set the value "Critical".
comment.setData("Severity", "Critical");

comment.getData("Author"); // returns "Mark"
comment.getData();         // returns {"Author":"Mark",
                             "Severity":"Critical"}
comment.getData("FooBar"); // returns undefined
```

`getDataKeys()` → {Array.<string>}

Gets an array of data keys known to this Comment.

See: [PCCViewer.Data#getDataKeys](#)

[PCCViewer.Comment#getData](#)

[PCCViewer.Comment#setData](#)

Returns:

Returns an array of data keys known to this Comment. If no data is stored, then an empty array will be returned.

Type

Array.<string>

Example

```
var comment = myConversation.addComment("Hello world.");

// Returns an empty array before key-value pairs are
// stored.
comment.getDataKeys(); // returns []

// Returns a list of all keys.
comment.setData("Author", "Mark");
comment.setData("Severity", "Critical");
comment.getDataKeys(); // returns ["Author", "Severity"]
```

[getMarkupLayer\(\)](#) → {[PCCViewer.MarkupLayer](#)}

Gets the markup layer that this comment is (or was) a part of.

Returns:

The comment's markup layer.

If a comment is removed from a markup layer, this still returns the markup layer object.

Type

[PCCViewer.MarkupLayer](#)

[getSessionData\(key\)](#) → {string|object}

Gets the session data value for the given key, or gets a hash containing all key values, if a key was not provided. Unlike [PCCViewer.Comment#getData](#), this data is not saved with the annotation, it only lasts for the session.

This method is defined on all Comment objects.

Parameters:

Name	Type	Description
key	string	The key for which to get the data value.

See: [PCCViewer.Data#getSessionData](#)
[PCCViewer.Comment#setSessionData](#)
[PCCViewer.Comment#getSessionDataKeys](#)

Throws:

If the `key` argument is null or otherwise not a string.

Type
Error

Returns:

- If a key argument was provided, it returns the associated value.
- If a key argument was provided, but a value has not been associated with the key, then it returns undefined.
- If a key was not provided, it returns a hash object containing all key-value pairs.

Type
string | object

Example

```
var comment = myConversation.addComment("Hello world.");

// The key "Author" is set the value "Mark".
comment.setSessionData("Author", "Mark");

// The key "Note" is set the value "This is not going to be
saved!".
comment.setSessionData("Note", "This is not going to be
saved!");

comment.getSessionData("Author"); // returns "Mark"
comment.getSessionData();        // returns
{"Author":"Mark", "Note":"This is not going to be saved!"}
comment.getSessionData("FooBar"); // returns undefined
```

`getSessionDataKeys()` → {Array.<string>}

Gets an array of data keys known to this Comment. Unlike [PCCViewer.Comment#getDataKeys](#), this data is not saved with the annotation, it only lasts for the session.

This method is defined on all Comment objects.

See: [PCCViewer.Data#getSessionDataKeys](#)
[PCCViewer.Comment#getSessionData](#)
[PCCViewer.Comment#setSessionData](#)

Returns:

Returns an array of data keys known to this Comment. If no data is stored, then an empty array will be returned.

Type
Array.<string>

Example

```
var comment = myConversation.addComment("Hello world.");

// Returns an empty array before key-value pairs are
// stored.
comment.getSessionDataKeys(); // returns []

// Returns a list of all keys.
comment.setSessionData("Author", "Mark");
comment.setSessionData("Note", "This is not going to be
saved!");
comment.getSessionDataKeys(); // returns ["Author", "Note"]
```

`getText()` → {string}

Gets the text content of the Comment.

Note: This content will be a plain text string which is not sanitized in any way. It may be necessary to sanitize and safely use the string when inserting it into the DOM.

See: [PCCViewer.Comment#setText](#)

Returns:

The Comment content.

Type
string

`setCreationTime(time)` → {PCCViewer.Comment}

Sets the date and time when the comment was created.

Parameters:

Name	Type	Description
time	Date string	A JavaScript <code>Date</code> Object or a string in the ISO 8601 format.

See: [PCCViewer.Comment#getCreationTime](#)

Throws:

- If the value is not a string or a `Date` object.

Type
Error

- If a string value is used that is not an ISO 8601 date format.

Type
Error

Returns:

The comment Object.

Type
[PCCViewer.Comment](#)

Example

```
// assume we already have a Comment
var now = Date.now();
comment.setCreationTime(now);

var janFirst1970 = "1970-01-01T00:00:00.000Z";
comment.setCreationTime(janFirst1970);
```

`setData(key, value)` → **`{PCCViewer.Comment}`**

Sets the data value for the given key.

Notes:

- Overwrites any data value already associated with the given key.
- There is no artificial limit imposed on the number of key-value pairs that are stored.
- If limits on the number of key-value pairs are required, they should be enforced by calling code.
- Setting the value as undefined results in no information for the key being persisted to the server.
- The returned data is not mutated or sanitized, which could lead to a security vulnerability if not sanitized properly before use.

Parameters:

Name	Type	Description
key	string	The key for which to set the data value.
value	string	This is the value to set for the key. <ul style="list-style-type: none">• This must be a string or undefined.• The maximum length of the string is not limited by this function.

See: [PCCViewer.Data#setData](#)
[PCCViewer.Comment#getData](#)
[PCCViewer.Comment#getDataKeys](#)

Returns:

Returns the Comment object on which the method was called.

Type

[PCCViewer.Comment](#)

Example

```
var comment = myConversation.addComment("Hello world.");

// Get data returns undefined before the key is set.
comment.getData("Author"); // returns undefined

// The key "Author" is set the value "Mark".
comment.setData("Author", "Mark");
comment.getData("Author"); // returns "Mark"

// The key "Author" is overwritten with the value "Clark".
comment.setData("Author", "Clark");
comment.getData("Author"); // returns "Clark"

// The key "Author" is unset, by setting the value to
// undefined.
comment.setData("Author", undefined);
comment.getData("Author"); // returns undefined

// The value can only be set to a string or undefined.
// All other data types throw.
comment.setData("FooBar", null); // throws
comment.setData("FooBar", 1); // throws
comment.setData("FooBar", true); // throws
comment.setData("FooBar", {}); // throws
comment.setData("FooBar", []); // throws
```

`setSessionData(key, value) → {PCCViewer.Comment}`

Sets the session data value for the given key. Unlike `PCCViewer.Comment#setData`, this data is not saved with the annotation, it only lasts for the session.

This method is defined on all Comment objects.

Parameters:

Name	Type	Description
key	string	The key for which to set the data value.
value	string	This is the value to set for the key. <ul style="list-style-type: none">• This must be a string or undefined.• The maximum length of the string is not limited by this function.

See: [PCCViewer.Data#setSessionData](#)
[PCCViewer.Comment#getSessionData](#)
[PCCViewer.Comment#getSessionDataKeys](#)

Returns:

The Comment object on which the method was called.

Type

[PCCViewer.Comment](#)

Example

```
var comment = myConversation.addComment("Hello world.");

// Get data returns undefined before the key is set.
comment.getSessionData("Author"); // returns undefined

// The key "Author" is set the value "Mark".
comment.setSessionData("Author", "Mark");
comment.getSessionData("Author"); // returns "Mark"

// The key "Author" is overwritten with the value "Clark".
comment.setSessionData("Author", "Clark");
comment.getSessionData("Author"); // returns "Clark"

// The key "Author" is unset, by setting the value to
// undefined.
comment.setSessionData("Author", undefined);
comment.getSessionData("Author"); // returns undefined

// The value can only be set to a string or undefined.
```

```
// All other data types throw.  
comment.setSessionData("FooBar", null); // throws  
comment.setSessionData("FooBar", 1);    // throws  
comment.setSessionData("FooBar", true); // throws  
comment.setSessionData("FooBar", {});   // throws  
comment.setSessionData("FooBar", []);   // throws
```

setText(text) → {PCCViewer.Comment}

Sets the text content of the Comment.

Parameters:

Name	Type	Description
text	string	The text content being set.

See: [PCCViewer.Comment#getText](#)

Returns:

The comment Object.

Type

[PCCViewer.Comment](#)

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: Conversation

PCCViewer. Conversation

new Conversation(mark)

A collection of comments associated with a specific [PCCViewer.Mark](#) Object.

A `Conversation` Object is already available on each [PCCViewer.Mark](#), and should not be created directly through this constructor.

Parameters:

Name	Type	Description
mark	PCCViewer.Mark	The Mark to which the conversation is attached.

Throws:

If `mark` is not a [PCCViewer.Mark](#) Object.

Type
Error

Example

```
// assume we already have some marks on the document
var firstMark = viewerControl.getAllMarks()[0];

var conversation = firstMark.getConversation();
```

Methods

`addComment(commentText)` → [{PCCViewer.Comment}](#)

Adds a comment to the `Conversation`.

Parameters:

Name	Type	Description
<code>commentText</code>	<code>string</code>	The text content of the comment being added.

See: [PCCViewer.Comment](#)

Throws:

If `commentText` is not a string.

Type
Error

Returns:

The newly created comment.

Type
[PCCViewer.Comment](#)

`deleteComments(comments)` → [{PCCViewer.Conversation}](#)

Deletes the specified comment or comments from the `Conversation`.

Parameters:

Name	Type	Description
comments	PCCViewer.Comment Array. < PCCViewer.Comment >	A comment or array of comments to be deleted.

Returns:

The conversation Object.

Type

[PCCViewer.Conversation](#)

[getComments\(\)](#) → {Array.<[PCCViewer.Comment](#)>}

Gets an array of all comments in the Conversation. The comments are ordered by the creation date and time.

Returns:

An array of Comments.

Type

Array.<[PCCViewer.Comment](#)>

[getData\(key\)](#) → {string|object}

Gets the data value for the given key, or gets a hash containing all key values, if a key was not provided.

Note: If a hash is returned, this will be a new object each time it is called. Adding new properties to the returned hash will not add data to the Conversation.

Note: The returned data is not mutated or sanitized, which could lead to a security vulnerability if not sanitized properly before use.

Parameters:

Name	Type	Description
key	string	The key for which to get the data value.

See:

[PCCViewer.Data#getData](#)

[PCCViewer.Conversation#setData](#)

[PCCViewer.Conversation#getDataKeys](#)

Throws:

If the `key` argument is null or otherwise not a string.

Type
Error

Returns:

- If a key argument was provided, it returns the associated value.
- If a key argument was provided, but a value has not been associated with the key, then it returns undefined.
- If a key was not provided, it returns a hash object containing all key-value pairs.

Type
string | object

Example

```
var conversation = myMark.getConversation();

// The key "Resolved" is set the value "false".
conversation.setData("Resolved", "false");

// The key "Severity" is set the value "Critical".
conversation.setData("Severity", "Critical");

conversation.getData("Resolved"); // returns "false"
conversation.getData();           // returns
{"Resolved":"false", "Severity":"Critical"}
conversation.getData("FooBar");  // returns undefined
```

`getDataKeys()` → {Array.<string>}

Gets an array of data keys known to this Conversation.

See: [PCCViewer.Data#getDataKeys](#)
[PCCViewer.Conversation#getData](#)
[PCCViewer.Conversation#setData](#)

Returns:

Returns an array of data keys known to this Conversation. If no data is stored, then an empty array will be returned.

Type
Array.<string>

Example


```
var conversation = myMark.getConversation();

// Returns an empty array before key-value pairs are
// stored.
conversation.getDataKeys(); // returns []

// Returns a list of all keys.
conversation.setData("Resolved", "false");
conversation.setData("Severity", "Critical");
conversation.getDataKeys(); // returns ["Resolved",
"Severity"]
```

getMark() → {[PCCViewer.Mark](#)}

Gets the [PCCViewer.Mark](#) to which the conversation is attached.

Returns:

The Mark to which the conversation is attached.

Type

[PCCViewer.Mark](#)

getSessionData(key) → {string|object}

Gets the session data value for the given key, or gets a hash containing all key values, if a key was not provided. Unlike [PCCViewer.Conversation#getData](#), this data is not saved with the annotation, it only lasts for the session.

This method is defined on all Conversation objects.

Parameters:

Name	Type	Description
key	string	The key for which to get the data value.

See: [PCCViewer.Data#getSessionData](#)
[PCCViewer.Conversation#setSessionData](#)
[PCCViewer.Conversation#getSessionDataKeys](#)

Throws:

If the `key` argument is null or otherwise not a string.

Type

Error

Returns:

- If a key argument was provided, it returns the associated value.
- If a key argument was provided, but a value has not been associated with the key, then it returns undefined.
- If a key was not provided, it returns a hash object containing all key-value pairs.

Type

string | object

Example

```
var conversation = myMark.getConversation();

// The key "Author" is set the value "Mark".
conversation.setSessionData("Author", "Mark");

// The key "Note" is set the value "This is not going to be
saved!".
conversation.setSessionData("Note", "This is not going to
be saved!");

conversation.getSessionData("Author"); // returns "Mark"
conversation.getSessionData();        // returns
{"Author":"Mark", "Note":"This is not going to be saved!"}
conversation.getSessionData("FooBar"); // returns undefined
```

getSessionDataKeys() → {Array.<string>}

Gets an array of data keys known to this Conversation. Unlike [PCCViewer.Conversation#getDataKeys](#), this data is not saved with the annotation, it only lasts for the session.

This method is defined on all Conversation objects.

See: [PCCViewer.Data#getSessionDataKeys](#)
[PCCViewer.Conversation#getSessionData](#)
[PCCViewer.Conversation#setSessionData](#)

Returns:

Returns an array of data keys known to this Conversation. If no data is stored, then an empty array will be returned.

Type

Array.<string>

Example

```
var conversation = myMark.getConversation();

// Returns an empty array before key-value pairs are
// stored.
conversation.getSessionDataKeys(); // returns []

// Returns a list of all keys.
conversation.setSessionData("Author", "Mark");
conversation.setSessionData("Note", "This is not going to
be saved!");
conversation.getSessionDataKeys(); // returns ["Author",
"Note"]
```

setData(key, value) → {PCCViewer.Conversation}

Sets the data value for the given key.

Notes:

- Overwrites any data value already associated with the given key.
- There is no artificial limit imposed on the number of key-value pairs that are stored.
- If limits on the number of key-value pairs are required, they should be enforced by calling code.
- Setting the value as undefined results in no information for the key being persisted to the server.
- The returned data is not mutated or sanitized, which could lead to a security vulnerability if not sanitized properly before use.

Parameters:

Name	Type	Description
key	string	The key for which to set the data value.
value	string	This is the value to set for the key. <ul style="list-style-type: none">• This must be a string or undefined.• The maximum length of the string is not limited by this function.

See: [PCCViewer.Data#setData](#)
[PCCViewer.Conversation#getData](#)
[PCCViewer.Conversation#getDataKeys](#)

Returns:

Returns the Conversation object on which the method was called.

Type

[PCCViewer.Conversation](#)

Example

```
var conversation = myMark.getConversation();

// Get data returns undefined before the key is set.
conversation.getData("Resolved"); // returns undefined

// The key "Resolved" is set the value "false".
conversation.setData("Resolved", "false");
conversation.getData("Resolved"); // returns "false"

// The key "Resolved" is overwritten with the value "true".
conversation.setData("Resolved", "true");
conversation.getData("Resolved"); // returns "true"

// The key "Resolved" is unset, by setting the value to
// undefined.
conversation.setData("Resolved", undefined);
conversation.getData("Resolved"); // returns undefined

// The value can only be set to a string or undefined.
// All other data types throw.
conversation.setData("FooBar", null); // throws
conversation.setData("FooBar", 1); // throws
conversation.setData("FooBar", true); // throws
conversation.setData("FooBar", {}); // throws
conversation.setData("FooBar", []); // throws
```

setSessionData(key, value) → {[PCCViewer.Conversation](#)}

Sets the session data value for the given key. Unlike [PCCViewer.Conversation#setData](#), this data is not saved with the annotation, it only lasts for the session.

This method is defined on all Conversation objects.

Parameters:

Name	Type	Description
key	string	The key for which to set the data value.
value	string	This is the value to set for the key. <ul style="list-style-type: none">• This must be a string or undefined.• The maximum length of the string is not limited by this function.

See: [PCCViewer.Data#setSessionData](#)
[PCCViewer.Conversation#getSessionData](#)
[PCCViewer.Conversation#getSessionDataKeys](#)

Returns:

The Conversation object on which the method was called.

Type

[PCCViewer.Conversation](#)

Example

```
var conversation = myMark.getConversation();

// Get data returns undefined before the key is set.
conversation.getSessionData("Author"); // returns undefined

// The key "Author" is set the value "Mark".
conversation.setSessionData("Author", "Mark");
conversation.getSessionData("Author"); // returns "Mark"

// The key "Author" is overwritten with the value "Clark".
conversation.setSessionData("Author", "Clark");
conversation.getSessionData("Author"); // returns "Clark"

// The key "Author" is unset, by setting the value to
// undefined.
conversation.setSessionData("Author", undefined);
conversation.getSessionData("Author"); // returns undefined

// The value can only be set to a string or undefined.
// All other data types throw.
conversation.setSessionData("FooBar", null); // throws
conversation.setSessionData("FooBar", 1); // throws
conversation.setSessionData("FooBar", true); // throws
conversation.setSessionData("FooBar", {}); // throws
conversation.setSessionData("FooBar", []); // throws
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: ConversionRequest

PCCViewer. ConversionRequest

(protected) `new ConversionRequest()`

The `ConversionRequest` object is created when converting a document using [PCCViewer.ViewerControl#requestDocumentConversion](#).

The `ConversionRequest` is a thenable object, which allows Promise-like interactions. Calling the [PCCViewer.ConversionRequest#then](#) method will return a [PCCViewer.Promise](#) object. On successful conversion, the `Promise` success method, if added, is called with an array of the urls to download the

converted documents. Note that it currently only supports converting to a single PDF file, so the output array will only contain a single URL. On a conversion failure, the `Promise` is rejected.

Note that if you create a `ConversionRequest` object using the [PCCViewer.ViewerControl#convertDocument](#) method, which has been marked as deprecated, the `Promise` success method is called with a single URL string (not an array of URL strings).

The `ConversionRequest` object also provides an event subscription method, to get notified of other types of information. See [PCCViewer.ConversionRequest.EventType](#).

Note: This constructor should not be used directly. Instead, a `ConversionRequest` is created by a call to [PCCViewer.ViewerControl#requestDocumentConversion](#).

Example

```
function onSuccessfullConvert(convertedDocumentUrls) {
    alert("The first converted document url: " +
    convertedDocumentUrls[0]);
}

function onFailedConvert(error) {
    alert("conversion process failed, reason: " +
    (error.message ? error.message : error));
}

// A ConversionRequest object is created by and returned
from the call to the
ViewerControl#requestDocumentConversion method
var conversionRequest =
viewerControl.requestDocumentConversion();
conversionRequest.then(onSuccessfulConvert,
onFailedConvert);

//register some events
conversionRequest

.on(PCCViewer.ConversionRequest.EventType.ConversionCompleted,

    function(ev) {
        alert("Document conversion completed.");
    })

.on(PCCViewer.ConversionRequest.EventType.ConversionProgress,

    function(event) {
        alert("Conversion progress: " + event.percent +
"%");
    })

.on(PCCViewer.ConversionRequest.EventType.ConversionFailed,
    function(event) {
        alert("Document conversion failed.");
    });
```

```
});
```

Members

(static, readonly) **EventType** :string

A list of events that can be triggered by the [PCCViewer.ConversionRequest](#) object.

Note: This enumeration is for convenience for API developers. Instead of using this enumeration, you can pass string values of the eventType (enumeration values)

Type:

- string

Properties:

Name	Description
ConversionProgress : string	Triggered when the conversion process receives an update. Augmented properties of the PCCViewer.Event object for this event: <ul style="list-style-type: none"> • <code>progress {number}</code> - Indicates the estimated percentage complete.
ConversionCompleted : string	Triggered when the conversion process completes successfully, fails to complete, or is cancelled. Augmented properties of the PCCViewer.Event object for this event: None
ConversionFailed : string	Triggered when the conversion process fails. Augmented properties of the PCCViewer.Event object for this event: None
ConversionCancelled : string	Triggered when user cancels the conversion process. Augmented properties of the PCCViewer.Event object for this event: None
ConversionWorkerCreated : string	Triggered when the conversion process is created by the backend conversion service. Augmented properties of the PCCViewer.Event object for this event: None

See: [PCCViewer.Event](#)
[PCCViewer.ConversionRequest#on](#)
[PCCViewer.ConversionRequest#off](#)

(readonly) **convertedDocumentDownloadURLs** :string|null

Gets the URLs for downloading each converted document after successful conversion.

This property is defined on all ConversionRequest objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string | null

See: [PCCViewer.ConversionRequest#getConvertedDocumentDownloadURLs](#)

Example

```
var downloadUrls =  
conversionRequest.convertedDocumentDownloadURLs;
```

(readonly) errorCode : number

Gets the errorCode if there is a failure during conversion process.

This property is defined on all ConversionRequest objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.ConversionRequest#getErrorCode](#)

Example

```
var errorCode = conversionRequest.errorCode;
```

(readonly) options :Object

Gets the options that were provided/used to convert the document.

This property is defined on all ConversionRequest objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.ConversionRequest#getOptions](#)

Example

```
var options = conversionRequest.options;
```

(readonly) **progress** :number

Gets the current estimate of the conversion process progress.

This property is defined on all ConversionRequest objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.ConversionRequest#getProgress](#)

Example

```
var percentProgress = conversionRequest.progress;
```

(readonly) **workerID** :number

Gets the worker ID.

This property is defined on all ConversionRequest objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.ConversionRequest#getWorkerID](#)

Example

```
var percentProgress = conversionRequest.workerID;
```

Methods

`cancel()` → `{PCCViewer.ConversionRequest}`

Cancels the conversion request. The `PCCViewer.Promise` object that is returned will be rejected. If the user cancels the operation, then a `PCCViewer.Error` object will be returned as the rejection reason and its code property will be set to `UserCancelled`.

Returns:

The cancelled request.

Type

`PCCViewer.ConversionRequest`

Example

```
var conversionRequest =
viewerControl.requestDocumentConversion();
conversionRequest.cancel();
```

`getConvertedDocumentDownloadURL()`

Deprecated since v11.0 (use the `PCCViewer.ConversionRequest#getConvertedDocumentDownloadURLs` method instead).

`getConvertedDocumentDownloadURLs()` → `{string|null}`

Gets the URLs for a web tier service to download each converted document.

Returns:

Returns an array of URLs to download the converted documents. If called before the process completes, or after the process fails, it will return `null`.

Type

`string | null`

Example

```
var conversionRequest =
viewerControl.requestDocumentConversion();
var convertedUrls =
conversionRequest.getConvertedDocumentDownloadURLs();
```

`getErrorCode()` → `{string}`

Gets the error code for the conversion request, in cases where the request has failed.

Returns:

A value for programmatic identification of an error condition, or null if an error has not occurred.

Type
string

Example

```
var conversionRequest =  
viewerControl.requestDocumentConversion();  
var errorCode = conversionRequest.getErrorCode();
```

getOptions() → {Object}

Gets a copy of the options object used by the conversion request.

The original options object has been provided to the method [PCCViewer.ViewerControl#requestDocumentConversion](#). If no options object was provided to `requestDocumentConversion`, or the options object did not define all properties, then the returned object will represent the actual options used.

Returns:

A copy of the conversion options object which is used by this conversion request.

- `targetExtension` {string} - (optional) Whether to convert the signatures.
- `filename` {string} - (optional) The format to which the document will be converted.

Type
Object

Example

```
var conversionRequest =  
viewerControl.requestDocumentConversion();  
var options = conversionRequest.getOptions();
```

getProgress() → {number}

Gets the currently known conversion progress value.

Returns:

A number between 0 and 100 (inclusive). A value of 100 means the conversion process completed.

Type

number

Example

```
var conversionRequest =
viewerControl.requestDocumentConversion();
var percent = conversionRequest.getProgress();
```

getWorkerID() → {string|null}

Gets the ID of the PCCIS ContentConverter performing conversion.

Returns:

The ID of the PCCIS ContentConverter performing conversion. Returns `null` before the worker is created.

Type

string | null

Example

```
var conversionRequest =
viewerControl.requestDocumentConversion();
var workerId = conversionRequest.getWorkerID();
```

off(eventType, handler) → {PCCViewer.ConversionRequest}

Unsubscribe an event handler from a specified event type.

Typically, an event is unsubscribed when you no longer want further notification of the event.

Parameters:

Name	Type	Description
eventType	string	A string specifying the event type. See PCCViewer.ConversionRequest.EventType for a list and description of all supported events.
handler	PCCViewer.Event~eventHandler	A function that was attached previously to the <code>ViewerControl</code> . Note: This must be the same function object previously passed to PCCViewer.ConversionRequest#on . It cannot be a different object that is functionally equivalent.

See: [PCCViewer.ConversionRequest#on](#)
[PCCViewer.ViewerControl#off](#) for more details on unsubscribing event handlers.

Returns:

The `ConversionRequest` object on which this method was called.

Type

[PCCViewer.ConversionRequest](#)

`on(eventType, handler) → {PCCViewer.ConversionRequest}`

Subscribe an event handler to an event of a specified type.

Parameters:

Name	Type	Description
<code>eventType</code>	<code>string</code>	A string that specifies the event type. This value is case-insensitive. See PCCViewer.ConversionRequest.EventType for a list and description of all supported events.
<code>handler</code>	PCCViewer.Event~eventHandler	A function that will be called whenever the event is triggered.

See: [PCCViewer.ConversionRequest#off](#)
[PCCViewer.ViewerControl#on](#) for more details on event subscription.

Returns:

The `ConversionRequest` object on which this method was called.

Type

[PCCViewer.ConversionRequest](#)

`then(onFulfilledopt, onRejectedopt) → {PCCViewer.Promise}`

Register callbacks to access the current or eventual result of the `ConversionRequest`.

On successful conversion, the `onFulfilled` callback(s) are called with the URL to download the converted document.

On a conversion failure, the `onRejected` callback(s) are called with a [PCCViewer.Error](#).

Multiple `onFulfilled` or `onRejected` callbacks can be registered for the same `ConversionRequest` by calling this method multiple times.

Parameters:

Name	Type	Attributes	Description
onFulfilled	PCCViewer.Promise~onFulfilled	<optional>	Called if or when the promise is resolved. Optionally pass a value of <code>null</code> or <code>undefined</code> if you do not use this callback, but you want to provide an <code>onRejected</code> callback.
onRejected	PCCViewer.Promise~onRejected	<optional>	Called if or when the promise is rejected.

Returns:

A promise object that is resolved according to the Promises/A+ standard.

Type

[PCCViewer.Promise](#)

Example

```
var viewerControl =
$("#myElement").pccViewer(...).viewerControl;

// a basic example
viewerControl.requestDocumentConversion().then(
  function onFulfilled(url) {
    // download document using `url`
  },
  function onRejected(error) {
    // Handle failure
  }
);
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: DocumentHyperlink

PCCViewer. DocumentHyperlink

The `PCCViewer.DocumentHyperlink` object represents hyperlinks in the original document.

There are two types of hyperlinks that can appear in a document, a `DocumentHyperlink` and a hyperlink drawn by an end user with the viewer's markup system, a "[Markup hyperlink](#)".

At a high level, these two different types of hyperlinks behave the same:

1. They can be clicked and the hyperlink followed.

There are several differences between the DocumentHyperlink and a Markup hyperlinks:

1. DocumentHyperlinks are written into the original document by the author of the original document. They are parsed out by the PrizmDoc Server and sent to the client viewer.
2. Markup hyperlinks are created by an end user of the viewer and saved and loaded with the rest of the markup in the viewer.
3. DocumentHyperlinks are immutable.
 - o Their attributes href, position, and styling cannot be modified.
 - o They cannot be deleted.
 - o They cannot be added.
4. Markup hyperlinks have full CRUD operation support via the API and mouse tools.
5. DocumentHyperlinks are loaded with the document.
6. Markup hyperlinks are loaded with the rest of the markup, which may be at the discretion of the end user or of the application embedding the PrizmDoc Viewing Client.

Constructor

`new DocumentHyperlink()`

NOTE: this constructor is for internal use only.

DocumentHyperlinks cannot be added via the API, they can only be retrieved via the [PCCViewer.ViewerControl#requestDocumentHyperlinks](#) method.

See: [PCCViewer.ViewerControl#requestDocumentHyperlinks](#)

Example

```
var viewerControl = new PCCViewer.ViewerControl(...);

// use
PCCViewer.ViewerControl#requestDocumentHyperlinks (pageNumber)

viewerControl.requestDocumentHyperlinks(1).then(
    function (documentHyperlinks) {
        // do something with the documentHyperlinks
        documentHyperlinks.forEach(function(dh) {
            // ...
        });
    },
    function (error) {
        alert("Something went wrong " + (error.message ?
error.message : error));
    }
);
```

Members

(readonly) `href` :string|number

Gets the link target for DocumentHyperlink.

Type:

- string | number

See: [PCCViewer.DocumentHyperlink#getHref](#)

Example

```
var href = documentHyperlink.href;

switch (typeof href) {
  case "number":
    // navigate to the page
    viewerControl.setPageNumber(href);
    break;
  case "string":
  default:
    // Interpret the URL and execute the navigation.
    window.location.href = href;
    break;
}
```

(readonly) **pageNumber** :number

Gets the page number where the DocumentHyperlink object is located.

Type:

- number

See: [PCCViewer.DocumentHyperlink#pageNumber](#)

Example

```
var pageNumber = myDocumentHyperlink.pageNumber;
```

(readonly) **rectangle** :number

Gets the bounding rectangle for the DocumentHyperlink. The returned object has the type {x: xValue, y: yValue, width: widthValue, height: heightValue}.

Type:

- number

See: [PCCViewer.DocumentHyperlink#rectangle](#)

Example

```
var boundingRectangle = myDocumentHyperlink.rectangle;
```

Methods

getHref() → {string|number}

Gets the link target for DocumentHyperlink.

See: [PCCViewer.DocumentHyperlink#href](#)

Returns:

The link target.

A number value indicates that the target is a page number within the document.

Type

string | number

Example

```
var href = documentHyperlink.getHref();

switch (typeof href) {
  case "number":
    // navigate to the page
    viewerControl.setPageNumber(href);
    break;
  case "string":
  default:
    // Interpret the URL and execute the navigation.
    window.location.href = href;
    break;
}
```

getPageNumber() → {number}

Gets the page number where the DocumentHyperlink object is located.

See: [PCCViewer.DocumentHyperlink#pageNumber](#)

Returns:

The page number where the DocumentHyperlink is located.

Type
number

Example

```
var pageNumber = myDocumentHyperlink.getPageNumber();
```

`getRectangle()` → {Object}

Gets the bounding rectangle for the DocumentHyperlink.

See: [PCCViewer.DocumentHyperlink#rectangle](#)

Returns:

A rectangle object of the type {x: xValue, y: yValue, width: widthValue, height: heightValue}.

Type
Object

Example

```
var boundingRectangle = myDocumentHyperlink.getRectangle();
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: Error

PCCViewer. Error

`new Error(code, message)`

The constructor for a `ERROR` Object. PCCViewer. Error inherits from the JavaScript Error.

Parameters:

Name	Type	Description
code	string	A string that indicates the error code. As a convention used across PrizmDoc Viewing Client and Server, the code should be PascalCased and should not contain any spaces.

Name	Type	Description
message	string	A developer readable string that indicates details of the error.

Example

```
// assume we already have a PCCViewer.Mark object created
var error = new PCCViewer.Error('ResponseEmpty',
  'Empty/invalid response from the server');
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: Event

PCCViewer. Event

new Event(target, type)

Create an event object. This is the internal constructor used when the viewer emits any event. This object describes the common attributes of all events. Augmented, event-specific, values can be found in the descriptions of each event. See [PCCViewer.EventType](#) for specific events.

Parameters:

Name	Type	Description
target	PCCViewer.ViewerControl	The event target is the viewer where the event originated.
type	string	The name of the event type.

Members

(readonly) **target** :[PCCViewer.ViewerControl](#)

Contains the instance of the viewer that fired the event. See also [PCCViewer.Event#getTarget](#).

ECMA5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- [PCCViewer.ViewerControl](#)

(readonly) **type** :string

Contains the type of event. See also [PCCViewer.Event#getType](#).

ECMA5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- string

Methods

getTarget() → {[PCCViewer.ViewerControl](#)}

Gets the instance of the viewer that fired the event.

Returns:

Type

[PCCViewer.ViewerControl](#)

Example

```
var viewerObj;
function pageCountReadyHandler (event) {
    console.log("page count ready event");
    if(event.getTarget() !== viewerObj){
        alert("The `pageCountReady` event did not originate
from the expected instance of the viewer object");
    }
}
//subscribe to the pageCountReady event
viewerObj = viewer.on(PCCViewer.EventType.PageCountReady,
pageCountReadyHandler);
```

getType() → {string}

Gets the name of the event type. This will be the same value as the `eventType` argument to the [PCCViewer.ViewerControl#on](#) function.

See: [PCCViewer.EventType](#) for event types.

Returns:

a string containing event type.

Type

string

Example

```
function pageCountReadyHandler (event) {
    console.log("page count ready event");
    if(event.getType() !==
PCCViewer.EventType.PageCountReady){
        alert("The event type did not match");
    }
}
//subscribe to the pageCountReady event
viewer.on(PCCViewer.EventType.PageCountReady,
pageCountReadyHandler);
```

Type Definitions

eventHandler(event)

The function to call when an event occurs. When the event is triggered, all subscribed event handlers are called.

Parameters:

Name	Type	Description
event	PCCViewer.Event	A PCCViewer.Event object that represents the event. The event object is often augmented with properties which provide event specific information.

See:

- [PCCViewer.ViewerControl#on](#)
- [PCCViewer.ViewerControl#off](#)
- [PCCViewer.SearchRequest#on](#)
- [PCCViewer.SearchRequest#off](#)
- [PCCViewer.PrintRequest#on](#)
- [PCCViewer.PrintRequest#off](#)

Example

```
// Our event handler declaration.
// The handler will be called with one argument of type
PCCViewer.Event.
function pageCountReadyEventHandler(event) {
    var target = event.getTarget(), // `getTarget` is
defined on every event object
    type = event.getType(); // `getType` is
defined on every event object

    // The PageCountReady event augments the event object
```

```
with property `pageCount`  
    var pageCount = event.pageCount;  
}  
  
// Subscribe the event handler to an event.  
viewerControl.on("PageCountReady",  
pageCountReadyEventHandler);
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: ImageStamps

PCCViewer. ImageStamps

new ImageStamps(object)

The constructor for a `ImageStamp` Object. 'ImageStamp' object represents APIs for requesting the images and source from the server.

Parameters:

Name	Type	Description
object	object	with link to imageHandler URL.

Example

```
var options = { imageHandlerUrl: "../pcc.ashx" }  
var stamp = PCCViewer.ImageStamps(options);
```

Methods

getImageSourceURL(imageStampId)

Returns the image source URL for the imageStampId

Parameters:

Name	Type	Description
imageStampId	string	

Example

```
var options = { imageHandlerUrl: "../pcc.ashx" }
var stampApi = new PCCViewer.ImageStamps(options);

var imageUrl = stampApi.getImageSourceURL(imageStampId);
```

requestImageSourceBase64()

Deprecated since v13.4 (use the [PCCViewer.ImageStamps#requestImageSourceBase64](#) method instead).

requestImageSourceBase64(imageStampId) → {PCCViewer.Promise}

Retrieves the data URL and data hash from the server for the provided image stamp ID. This method utilizes an asynchronous server request to fetch the data and returns a [PCCViewer.Promise](#) to resolve the request.

The [onFulfilled](#) callback will receive an object containing the `dataUrl` (a base64 encoded image source) and `dataHash` (an encoded unique ID) of the image stamp.

If unable to retrieve the image stamp data URL and data hash from the server, then the returned [PCCViewer.Promise](#) object is rejected with the reason set to a [PCCViewer.Error](#) object with its code property set to `ImageStampDataFail`.

If AJAX is not supported, then the returned [PCCViewer.Promise](#) object is rejected with the reason set to a [PCCViewer.Error](#) object with its code property set to `AjaxUnsupported`.

If a server error is encountered, then the returned [PCCViewer.Promise](#) object is rejected with the reason set to a [PCCViewer.Error](#) object with its code property set to `Error`.

Parameters:

Name	Type	Description
<code>imageStampId</code>	string	A unique ID that corresponds to an image stamp stored on the server.

Returns:

A [PCCViewer.Promise](#) object.

Type

[PCCViewer.Promise](#)

Example

```
var options = { imageHandlerUrl: "../pcc.ashx" };
var stampApi = new PCCViewer.ImageStamps(options);

var stampPromise = stampApi.requestImageStampList();

stampPromise.then(
  function onSuccess(data) {
    var base64Promise =
```

```
stampApi.requestImageSourceBase64(data.imageStamps[0].id);

    base64Promise.then(
        function onSuccess(imageSource) {
            var pageNumber = 1;
            var rectangle = { x: 30, y: 30, width: 100,
height: 100 };
            var mark =
viewer.viewerControl.addMark(pageNumber,
"ImageStampAnnotation");
            mark.setImage({
                dataUrl: imageSource.dataUrl, id:
imageSource.dataHash
            });
            mark.setRectangle(rectangle);
        }
    );

},
function onFailure(error) {
    alert(error.message ? error.message : error);
}
);
```

requestImageStampList(object) → {PCCViewer.Promise}

Retrieves the image stamp list from the server.

If a server error is encountered, then the returned PCCViewer.Promise object is rejected with the reason set to a PCCViewer.Error object with its code property set to Error.

If AJAX is not supported, then the returned PCCViewer.Promise object is rejected with the reason set to a PCCViewer.Error object with its code property set to AjaxUnsupported.

Parameters:

Name	Type	Description
object	object	with web handler link assigned to the imageHandlerUrl property.

Returns:

a Promise object.

Type

[PCCViewer.Promise](#)

Example

```
var options = { imageHandlerUrl: "../pcc.ashx" }
```



```
var stampApi = new PCCViewer.ImageStamps(options);
var imageStamps = {};
var base64Source = null;

var stampPromise = stampApi.requestImageStampList();

stampPromise.then(
  function onSuccess(data) {
    imageStamps = data.imageStamps;
    var base64Promise =
stampApi.requestImageSourceBase64(imageStamps[0].id);

    base64Promise.then(
      function onSuccess(imageSource) {
        base64Source = imageSource
        var pageNumber = 1;
        var rectangle = { x: 30, y: 30, width:
100, height: 100 };
        var mark =
viewer.viewerControl.addMark(pageNumber,
"ImageStampAnnotation");
        mark.setImage({
          dataUrl: base64Source.dataUrl, id:
base64Source.dataHash
        });
        mark.setRectangle(rectangle);
      }
    );
  },
  function onFailure(error) {
    alert(error.message ? error.message : error)
  }
);
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: LoadMarkupLayersRequest

PCCViewer. LoadMarkupLayersRequest

(protected) **new** [LoadMarkupLayersRequest\(viewerControl, layerRecordIds\)](#)

The [LoadMarkupLayersRequest](#) object is created when loading markup layers using [PCCViewer.ViewerControl#loadMarkupLayers](#).

The [LoadMarkupLayersRequest](#) is a thenable object, which allows Promise-like interactions. Calling the [PCCViewer.LoadMarkupLayersRequest#then](#) method will return a [PCCViewer.Promise](#) object. On successful

layer loading, the `Promise` success method, if added, is called with the `PCCViewer.LoadMarkupLayer` objects created by the loaded layer records. On a load failure, the `Promise` is rejected.

The `LoadMarkupLayersRequest` object also provides an event subscription method, to get notified of other types of information. See [PCCViewer.LoadMarkupLayersRequest.EventType](#).

Note: This constructor should not be used directly. Instead, a `LoadMarkupLayersRequest` is created by a call to `PCCViewer.ViewerControl#loadMarkupLayers`.

Parameters:

Name	Type	Description
<code>viewerControl</code>	<code>string</code>	The <code>PCCViewer.ViewerControl</code> for the loaded document.
<code>layerRecordIds</code>	<code>Array.<string></code>	An array of layer record IDs.

Example

```
function onSuccessfullLoad(annotationLayers) {
    console.log(annotationLayers);
}

function onFailedConvert(error) {
    alert("Markup layer record loading failed, reason:" +
        (error.message ? error.message : error));
}

// A LoadMarkupLayersRequest object is created by and
// returned from the call to the ViewerControl#convertDocument
// method
var LoadMarkupLayersRequest =
viewerControl.loadMarkupLayers(['abc123']);
LoadMarkupLayersRequest.then(onSuccessfulConvert,
onFailedConvert);

//register some events
LoadMarkupLayersRequest

.on(PCCViewer.LoadMarkupLayersRequest.EventType.LoadMarkupLayers

    function(ev) {
        alert("Markup layer loading completed.");
    })

.on(PCCViewer.LoadMarkupLayersRequest.EventType.LoadMarkupLayers

    function(event) {
        alert("Conversion progress: " + event.percent +
"%");
    })
```

```

.on(PCCViewer.LoadMarkupLayersRequest.EventType.LoadMarkupLayersFailed)

    function(event) {
        alert("Markup layer loading failed.");
    }

.on(PCCViewer.LoadMarkupLayersRequest.EventType.LoadMarkupLayersCancelled)

    function(event) {
        alert("Markup layer loading was cancelled.");
    });

```

Members

(static, readonly) **EventType** :string

A list of events that can be triggered by the [PCCViewer.LoadMarkupLayersRequest](#) object.

Note: This enumeration is for convenience for API developers. Instead of using this enumeration, you can pass string values of the eventType (enumeration values)

Type:

- string

Properties:

Name	Description
LoadMarkupLayersFailed : string	
LoadMarkupLayersCompleted : string	
LoadMarkupLayersCancelled : string	
LoadMarkupLayersProgress : string	

See: [PCCViewer.Event](#)
[PCCViewer.LoadMarkupLayersRequest#on](#)
[PCCViewer.LoadMarkupLayersRequest#off](#)

(readonly) **errorCode** :number

Gets the errorCode if there is a failure during load process.

This property is defined on all LoadMarkupLayersRequest objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.LoadMarkupLayersRequest#getErrorCode](#)

Example

```
var errorCode = LoadMarkupLayersRequest.errorCode;
```

(readonly) **progress** :number

Gets the current estimate of the loading process progress.

This property is defined on all LoadMarkupLayersRequest objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.LoadMarkupLayersRequest#getProgress](#)

Example

```
var percentProgress = LoadMarkupLayersRequest.progress;
```

(readonly) **viewerControl** :Object

Gets the viewer control associated with this request.

This property is defined on all LoadMarkupLayersRequest objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.LoadMarkupLayersRequest#getViewerControl](#)

Example

```
var viewerControl = LoadMarkupLayersRequest.viewerControl;
```

Methods

`cancel()` → `{PCCViewer.LoadMarkupLayersRequest}`

Cancels the load request. The `PCCViewer.Promise` object that is returned will be rejected. If the user cancels the operation, then a `PCCViewer.Error` object will be returned as the rejection reason and its `code` property will be set to `UserCancelled`.

Returns:

The cancelled request.

Type

`PCCViewer.LoadMarkupLayersRequest`

Example

```
var loadMarkupLayersRequest =
viewerControl.loadMarkupLayers(['abc123']);
loadMarkupLayersRequest.cancel();
```

`getErrorCode()` → `{string}`

Gets the error code for the load request, in cases where the request has failed.

Returns:

A value for programmatic identification of an error condition, or null if an error has not occurred.

Type

string

Example

```
var loadMarkupLayersRequest =
viewerControl.loadMarkupLayers(['abc123']);
var errorCode = loadMarkupLayersRequest.getErrorCode();
```

`getProgress()` → `{number}`

Gets the currently known loading progress value.

Returns:

A number between 0 and 100 (inclusive). A value of 100 means the loading process completed.

Type
number

Example

```
var loadMarkupLayersRequest =
viewerControl.loadMarkupLayers(['abc123']);
var percent = loadMarkupLayersRequest.getProgress();
```

getViewerControl() → {[PCCViewer.ViewerControl](#)}

Gets the viewer control associated with this request.

Returns:

A viewer control object.

Type
[PCCViewer.ViewerControl](#)

Example

```
var loadMarkupLayersRequest =
viewerControl.loadMarkupLayers(['abc123']);
var viewerControl =
loadMarkupLayersRequest.getViewerControl();
```

off(eventType, handler) → {[PCCViewer.LoadMarkupLayersRequest](#)}

Unsubscribe an event handler from a specified event type.

Typically, an event is unsubscribed when you no longer want further notification of the event.

Parameters:

Name	Type	Description
eventType	string	A string specifying the event type. See PCCViewer.LoadMarkupLayersRequest.EventType for a list and description of all supported events.
handler	PCCViewer.Event~eventHandler	A function that was attached previously to the <code>ViewerControl</code> . Note: This must be the same function object previously passed to PCCViewer.LoadMarkupLayersRequest#on . It cannot be a different object that is functionally equivalent.

See: [PCCViewer.LoadMarkupLayersRequest#on](#)
[PCCViewer.ViewerControl#off](#) for more details on unsubscribing event handlers.

Returns:

The `LoadMarkupLayersRequest` object on which this method was called.

Type
[PCCViewer.LoadMarkupLayersRequest](#)

`on(eventType, handler) → {PCCViewer.LoadMarkupLayersRequest}`

Subscribe an event handler to an event of a specified type.

Parameters:

Name	Type	Description
eventType	string	A string that specifies the event type. This value is case-insensitive. See PCCViewer.LoadMarkupLayersRequest.EventType for a list and description of all supported events.
handler	PCCViewer.Event~eventHandler	A function that will be called whenever the event is triggered.

See: [PCCViewer.LoadMarkupLayersRequest#off](#)
[PCCViewer.ViewerControl#on](#) for more details on event subscription.

Returns:

The `LoadMarkupLayersRequest` object on which this method was called.

Type
[PCCViewer.LoadMarkupLayersRequest](#)

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: Mark

PCCViewer. Mark

The `PCCViewer.Mark` object represents an annotation or redaction. (We use the term "mark" as a generic term for annotations and redactions.)

Marks Drawn on the Document

Typically a Mark object represents a mark that is drawn on the document. In the case that a Mark object represents a mark drawn on the document, then setting properties of the Mark object will cause the appearance of the mark on the document to be updated.

To add a new mark to a document, use the method [PCCViewer.ViewerControl#addMark](#).

Mark Properties

All marks have the following **read-only properties** that are set when the API creates the Mark object.

- `.getId()` and `.id`
- `.getType()` and `.type`
- `.getPageNumber()` and `.pageNumber`

A Mark object can also have a number of **writable properties**, which are define on the object depending on the mark type.

For example, a Mark object with type `LineAnnotation` will have a property `.startPoint`, accessible through `.getStartPoint()` and `.setStartPoint(...)`, but a Mark object with the type `RectangleAnnotation` will not have the property `.startPoint`. Or, a Mark object with the type `StampAnnotation` will have a property `.color`, accessible through `.getColor()` and `.setColor(...)` and a `LineAnnotation` Mark object also has the property `.color`.

The full list of Mark properties is listed below. The documentation lists what properties are on what mark types, and vice versa.

To programmatically determine if a mark has a property, use the JavaScript `in` operator. Or use the `!` operator if you are referencing the property's getter or setter method. Examples are shown below.

```
!!myMarkObject.getStartPoint; // true if the startPoint property was
added to the Mark object
//Note: this option will only be available in ECMAScript 5 compatible
browsers
'startPoint' in myMarkObject; // true if the startPoint property was
added to the Mark object
```

Template Marks

A Mark object can also act as a template for mouse tools that create marks. For example, a `LineAnnotation` mouse tool will have a "template mark" that is used as a template for any mark created by the mouse tool. Setting the color of the template mark to yellow will cause all marks created by the mouse tool to be yellow.

The template mark is accessible via the method [PCCViewer.MouseTool#getTemplateMark](#).

Constructor

`new Mark(parameters)`

NOTE: this constructor is for internal use only.

To programmatically add a Mark (an annotation or redaction) to a document, use [PCCViewer.ViewerControl#addMark](#).

Parameters:

Name	Type	Description
parameters	Object	The parameters object takes the following properties: <ul style="list-style-type: none"> • <code>type</code> {string} [required] Mark Type PCCViewer.Mark.Type • <code>pageNumber</code> {number} [optional] Page Number on which the Mark is located . If this property is not specified, the page number will default to the current page.

See: [PCCViewer.ViewerControl#addMark](#)
[PCCViewer.MouseTool#getTemplateMark](#)
[PCCViewer.Mark.Type](#)

Example

```
var viewerControl = new PCCViewer.ViewerControl(...);

// use PCCViewer.ViewerControl#addMark(pageNumber,
// markType) to create
// and add a mark instead of new Mark()
viewerControl.addMark(1, "LineAnnotation");
```

Members

(static, readonly) **FontStyles** :string

The `PCCViewer.Mark.FontStyles` enumeration defines Font Styles known to the `ViewerControl`.

Note: This enumeration is for convenience for API developers. Instead of using this enumeration, you may pass string values of the `FontStyles` (enumeration values).

Type:

- string

Properties:

Name	Description
Bold : string	Specifies that the text should be bold.
Italic : string	Specifies that the text should be italic.
Strikeout : string	Specifies that the text should be struck out.
Underline : string	Specifies that the text should be underlined.

See: [PCCViewer.Mark#setFontStyle](#)

Example

```
// use the enumeration to make text Bold
mark.setFontStyle([PCCViewer.Mark.FontStyles.Bold]);

// or just use the string value. Note: The parameter
// should be an array of styles required.
mark.setFontStyle(["Bold"]);
```

(static, readonly) HorizontalAlignment :string

The `PCCViewer.Mark.HorizontalAlignment` enumeration defines horizontal alignment known to the `ViewerControl`.

Note: This enumeration is for convenience for API developers. Instead of using this enumeration, you may pass string values of the `HorizontalAlignment` (enumeration values).

Type:

- string

Properties:

Name	Description
Center : string	Specifies that the text should be centered horizontally.
Left : string	Specifies that the text should be left-justified.
Right : string	Specifies that the text should be right-justified.

Example

```
// use the enumeration to make text centered
mark.setHorizontalAlignment(PCCViewer.Mark.HorizontalAlignment.Center);

// or just use the string value
mark.setHorizontalAlignment("Center");
```

(static, readonly) InteractionMode :string

The `PCCViewer.Mark.InteractionMode` enumeration defines possible interaction modes for a `Mark`. The interaction mode affects the behavior of the mark when the user interacts with the mark through mouse, touch, or API.

Note: This enumeration is for convenience for API developers. Instead of using this enumeration, you may pass string values of the `InteractionMode` (enumeration values).

Type:

- string

Properties:

Name	Description
<code>Full</code> : string	Specifies that the mark is fully interactive using the mouse, touch-input, and API.
<code>SelectionDisabled</code> : string	Specifies that the mark cannot be selected. Note: If a mark is selected when the mark's interaction mode is set to "SelectionDisabled", then the mark will be deselected and the ViewerControl's MarkSelectionChanged event will fire.

See: [PCCViewer.Mark#getInteractionMode](#)
[PCCViewer.Mark#setInteractionMode](#)
[PCCViewer.Mark#interactionMode](#)

Example

```
// use the enumeration to make the mark non-selectable
mark.setInteractionMode(PCCViewer.Mark.InteractionMode.SelectionDisabled);

// or just use the string value of the enumeration
mark.setInteractionMode("SelectionDisabled");
```

(static, readonly) `LineHeadType` :string

The `PCCViewer.Mark.LineHeadType` enumeration defines Mark LineHeadTypes known to the ViewerControl.

Note: This enumeration is for convenience for API developers. Instead of using this enumeration, you may pass string values of the `LineHeadType` (enumeration values).

Type:

- string

Properties:

Name	Description
<code>None</code> : string	No line head.
<code>FilledTriangle</code> : string	A filled triangle line head. Note: this makes the line an arrow. :)

Example

```
// use the enumeration to make a line an arrow
myLineAnnotation.setEndHeadType(PCCViewer.Mark.LineHeadType.

// or just use the string value
myLineAnnotation.setEndHeadType("FilledTriangle");
```

(static, readonly) **Type** :string

The `PCCViewer.Mark.Type` enumeration defines Mark Types known to the `ViewerControl`.

Note: This enumeration is for convenience for API developers. Instead of using this enumeration, you may pass string values of the `Type` (enumeration values).

Type:

- string

Properties:

Name	Description
LineAnnotation : string	This mark is drawn as a line on the document. A head can be added to the end of the line to make the line an arrow. The properties specific for this are: <code>color</code> , <code>thickness</code> , <code>opacity</code> , <code>startPoint</code> , <code>endPoint</code> , <code>endHeadType</code> .
RectangleAnnotation : string	This mark is drawn as a rectangle on the document. The properties specific for this are: <code>fillColor</code> , <code>opacity</code> , <code>borderColor</code> , <code>borderThickness</code> , <code>rectangle</code> .
EllipseAnnotation : string	This mark is drawn as an ellipse on the document. The properties specific for this are: <code>fillColor</code> , <code>opacity</code> , <code>borderColor</code> , <code>borderThickness</code> , <code>rectangle</code> .
TextAnnotation : string	This mark is drawn as text on the document. The text has a background and border. The properties specific for this are: <code>text</code> , <code>fontColor</code> , <code>fillColor</code> , <code>opacity</code> , <code>borderColor</code> , <code>borderThickness</code> , <code>fontName</code> , <code>fontSize</code> , <code>fontStyle</code> , <code>horizontalAlignment</code> , <code>rectangle</code> . The methods specific for this are: <code>highlightText</code> .
StampAnnotation : string	This mark is drawn as a stamp on the document. A stamp has a label (text) and a border. The properties specific for this are: <code>color</code> , <code>label</code> , <code>rectangle</code> .
HighlightAnnotation : string	This mark is drawn as a text highlight on the document.

Name	Description
	<p>The properties specific for this are: <code>fillColor</code> and <code>text</code>.</p> <p>The methods specific for this are: <code>highlightText</code>.</p>
RectangleRedaction : string	<p>This mark is drawn as a rectangle on the document.</p> <p>The properties specific for this are: <code>rectangle</code>, <code>borderColor</code>, <code>borderThickness</code>, <code>fillColor</code>, <code>fontColor</code>, <code>reason</code>, <code>reasons</code>. Note: You can provide either a plural <code>reasons</code> property with an array of strings or a singular <code>reason</code> property with a single string value. If you provide an array of <code>reasons</code>, they will be displayed together as a single string with a semicolon separating each reason.</p>
TransparentRectangleRedaction : string	<p>This mark is drawn as a transparent rectangle on the document. The color is always yellow and the opacity is always 50% (127/255).</p> <p>The properties specific for this are: <code>rectangle</code>.</p>
TextHyperlinkAnnotation : string	<p>This mark is drawn as a text hyperlink on the document.</p> <p>The properties specific for this are: <code>href</code>, <code>position</code> and <code>text</code> (read-only).</p> <p>The methods specific for this are: <code>highlightText</code>.</p>
TextRedaction : string	<p>This mark is drawn as a rectangle on the document. This redaction can be burned into the document.</p> <p>The properties specific for this are: <code>text</code>, <code>fontColor</code>, <code>fontName</code>, <code>fontSize</code>, <code>fontStyle</code>, <code>rectangle</code>.</p> <p>The methods specific for this are: <code>highlightText</code>.</p>
StampRedaction : string	<p>This mark is drawn as a stamp on the document. This redaction can be burned into the document.</p> <p>The properties specific for this are: <code>label</code>, <code>rectangle</code>.</p>
FreehandAnnotation : string	<p>This mark is drawn as a freehand line on the document.</p> <p>The properties specific for this are: <code>path</code>, <code>rectangle</code>, <code>color</code>, <code>thickness</code>, <code>opacity</code>.</p>
FreehandSignature : string	<p>This mark is drawn as a signature on the document and is confined to a rectangle.</p> <p>The properties specific for this are: <code>path</code>, <code>rectangle</code>, <code>color</code>, <code>thickness</code>.</p>
TextSignature : string	<p>This mark is drawn as a text signature on the document and is confined to a rectangle.</p> <p>The properties specific for this are: <code>text</code>, <code>rectangle</code>, <code>color</code>, <code>fontName</code>.</p>
TextInputSignature : string	<p>This mark is drawn as a single line of text that auto-fits to the containing rectangle. The user can interact with the mark using</p>

Name	Description
	<p>the mouse, touch-screen, and keyboard in order to set the text and adjust the rectangle.</p> <p>The properties specific for this are: <code>text</code>, <code>rectangle</code>, <code>fontColor</code>, <code>fontName</code>, <code>mask</code>, <code>horizontalAlignment</code>, <code>maxLength</code>.</p>
<code>TextAreaSignature</code> : string	<p>This mark is drawn as a text area that auto-fits to the containing rectangle. The user can interact with the mark using the mouse, touch-screen, and keyboard in order to set the text and adjust the rectangle.</p> <p>The properties specific for this are: <code>text</code>, <code>rectangle</code>, <code>fontColor</code>, <code>fontName</code>, <code>maxFontSize</code>, <code>fontStyle</code>, <code>horizontalAlignment</code>, <code>maxLength</code>.</p>
<code>TextSelectionRedaction</code> : string	<p>This mark is drawn as a text highlight redaction on the document.</p> <p>The properties specific for this are: <code>text</code>, <code>reason</code>, <code>reasons</code>. Note: You can provide either a plural <code>reasons</code> property with an array of strings or a singular <code>reason</code> property with a single string value. If you provide an array of <code>reasons</code>, they will be displayed together as a single string with a semicolon separating each reason.</p> <p>The methods specific for this are: <code>highlightText</code>.</p>
<code>ImageStampAnnotation</code> : string	<p>This mark is drawn as an ImageStamp annotation on the document.</p> <p>The properties specific for this are: <code>none</code>.</p>
<code>ImageStampRedaction</code> : string	<p>This mark is drawn as an ImageStamp redaction on the document.</p> <p>The properties specific for this are: <code>none</code>.</p>
<code>PolylineAnnotation</code> : string	<p>This mark is drawn as a Polyline on the document. It is in a form of a set of connected line segments.</p> <p>The properties specific for this are: <code>color</code>, <code>thickness</code>, <code>opacity</code>, <code>points</code>.</p>
<code>StrikethroughAnnotation</code> : string	<p>This mark is drawn as a Strikethrough annotation on the document. The annotation itself is just a line that is placed over the specified text.</p> <p>The properties specific for this are: <code>color</code>, <code>thickness</code> and <code>text</code>.</p> <p>The methods specific for this are: <code>highlightText</code>.</p>

Example

```
// use the enumeration to make a line annotation
myViewerControl.addMark(1,
PCCViewer.Mark.Type.LineAnnotation);

// or just use the string value
myViewerControl.addMark(1, "LineAnnotation");
```

borderColor :string

Gets or sets the border color of the Mark.

This property is defined on marks of type: `RectangleAnnotation`, `EllipseAnnotation`, `TextAnnotation` and `RectangleRedaction`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getBorderColor](#)
[PCCViewer.Mark#setBorderColor](#)

Example

```
if ('borderColor' in mark) {
    var oldValue = mark.borderColor;
    mark.borderColor = "#FF0000"; // set border color to
red
}
```

borderThickness :number

Gets or sets the border thickness of the Mark.

This property is defined on marks of type: `RectangleAnnotation`, `EllipseAnnotation`, `TextAnnotation` and `RectangleRedaction`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.Mark#getBorderThickness](#)
[PCCViewer.Mark#setBorderThickness](#)

Example

```
if ('borderThickness' in mark) {
    var oldValue = mark.borderThickness;
    mark.borderThickness = 3; // set the border thickness
of the mark
}
```

boundingRectangle :Object

Gets the bounding rectangle of the Mark.

This property is defined on all Mark objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.Mark#getBoundingBox](#)

Example

```
var boundingRectangle = mark.boundingBox;
```

color :string

Gets or sets the color of the Mark.

This property is defined on marks of type: `LineAnnotation`, `StampAnnotation`, `FreehandAnnotation`, `FreehandSignature`, `TextSignature`, `PolylineAnnotation` and `StrikethroughAnnotation`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getColor](#)
[PCCViewer.Mark#setColor](#)

Example

```
if ('color' in mark) {
```



```
var oldValue = mark.color;
mark.color = "#FF0000"; // set color to red
}
```

(readonly) conversation :string

Gets the conversation associated with the Mark.

This property is defined on all Mark objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getConversation](#)

Example

```
var conversation = mark.conversation;
```

endHeadType :string

Gets or sets the end head type of the Mark.

The value of end head type determines if the line looks like an arrow or a plain line.

This property is defined on marks of type: `LineAnnotation`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark.LineHeadType](#)
[PCCViewer.Mark#getEndHeadType](#)
[PCCViewer.Mark#setEndHeadType](#)

Example

```
if ('endHeadType' in mark) {
    var oldValue = mark.endHeadType;

    // set the head to be a filled triangle, so the line
    looks like an arrow
}
```

```
mark.endHeadType = "FilledTriangle";  
}
```

endPoint :Object

Gets or sets the end point coordinates of the line Mark.

This property is defined on marks of type: `LineAnnotation`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.Mark#getEndPoint](#)
[PCCViewer.Mark#setEndPoint](#)

Example

```
if ('endPoint' in mark) {  
    var oldValue = mark.endPoint;  
    mark.endPoint = {x: 100, y: 100}; // set the start  
    point to (100, 100)  
}
```

endPoint :Object

Gets or sets the end point coordinates of the line Mark.

This property is defined on marks of type: `LineAnnotation`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.Mark#getEndPoint](#)
[PCCViewer.Mark#setEndPoint](#)

Example

```
if ('endPoint' in mark) {  
    var oldValue = mark.endPoint;  
    mark.endPoint = {x: 100, y: 100}; // set the start
```

```
point to (100, 100)
}
```

fillColor :string

Gets or sets the fill color of the Mark.

This property is defined on marks of type: `RectangleAnnotation`, `EllipseAnnotation`, `TextAnnotation`, `HighlightAnnotation`, `'TextHyperlinkAnnotation'` and `RectangleRedaction`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getFillColor](#)
[PCCViewer.Mark#setFillColor](#)

Example

```
if ('fillColor' in mark) {
    var oldValue = mark.fillColor;
    mark.fillColor = "#FF0000"; // set color to red
}
```

fontColor :string

Gets or sets the font color of the text in the Mark.

This property is defined on marks of type: `TextAnnotation`, `TextRedaction`, `TextInputSignature`, `RectangleRedaction` and `TextAreaSignature`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getFontColor](#)
[PCCViewer.Mark#setFontColor](#)

Example

```
if ('fontColor' in mark) {
    var oldValue = mark.fontColor;
```

```
mark.fontColor = "#ffccbb";  
}
```

fontName :string

Gets or sets the font name of the text in the Text Mark.

This property is defined on marks of type: `TextAnnotation`, `TextRedaction`, `TextSignature`, `TextAreaSignature`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getFontName](#)
[PCCViewer.Mark#setFontName](#)

Example

```
if ('fontName' in mark) {  
    var oldValue = mark.fontName;  
    mark.fontName = "Aerial";  
}
```

fontSize :number

Gets or sets the font size (in points) for the text in the Text Mark.

This property is defined on marks of type: `TextAnnotation`, `TextRedaction`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.Mark#getFontName](#)
[PCCViewer.Mark#setFontName](#)

Example

```
if ('fontSize' in mark) {  
    var oldValue = mark.fontSize;  
    mark.fontSize = 12;  
}
```

```
}
```

fontStyle :Array.<string>

Gets or sets the font Style of the text in the mark.

This property is defined on marks of type: `TextRedaction`, `TextAreaSignature`, `TextAnnotation`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Array.<string>

See: [PCCViewer.Mark.FontStyles](#)
[PCCViewer.Mark#getFontStyle](#)
[PCCViewer.Mark#setFontStyle](#)

Example

```
if ('fontStyle' in mark) {  
    var oldValue = mark.fontStyle;  
    mark.fontStyle = ["Bold", Underline, Italic];  
}
```

horizontalAlignment :string

Gets or sets the text horizontal alignment of the text in the Text Mark.

This property is defined on marks of type: `TextAnnotation`, `TextRedaction`, `FreehandSignature`, `TextSignature`, `TextInputSignature` and `TextAreaSignature`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark.HorizontalAlignment](#)
[PCCViewer.Mark#getHorizontalAlignment](#)
[PCCViewer.Mark#setHorizontalAlignment](#)

Example

```
if ('horizontalAlignment' in mark) {
```

```
var oldValue = mark.horizontalAlignment;
mark.horizontalAlignment = "left";
}
```

href :string

Gets or sets the link target for hyperlink annotations.

All strings and numbers are valid values. It is the responsibility of the API consumer to handle clicks of hyperlink annotations. When handling the click, the API consumer should interpret the href value and take the appropriate navigation action.

This property is defined on marks of type: `TextHyperlinkAnnotation`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getHref](#)
[PCCViewer.Mark#setHref](#)

Example

```
if ('href' in mark) {
    var oldValue = mark.href;
    mark.href = "http://www.accusoft.com/";
}
```

(readonly) id :string

Gets the ID of the Mark.

This property is defined on all Mark objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getId](#)

Example

```
var id = mark.id;
```

`image` : [PCCViewer.Mark~ImageData](#)

Gets the image that is displayed for the Mark.

This property is defined on marks of type: `ImageStampAnnotation` and `ImageStampRedaction`

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- [PCCViewer.Mark~ImageData](#)

See: [PCCViewer.Mark#getImage](#)
[PCCViewer.Mark#setImage](#)

Example

```
if ('image' in mark) {
    // get old image data
    var oldImageData = mark.image;

    // set new image
    mark.image = {
        dataUrl: "data:image/png;base64,base64 string",
        id: "myUniqueImageIDABC123"
    };
}
```

`interactionMode` :string

Gets or sets a value that indicates the allowed interactions with the mark. Possible values are defined in the enumeration [PCCViewer.Mark.InteractionMode](#).

This property is defined on all Mark objects.

This is an ECMA 5 property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getInteractionMode](#)
[PCCViewer.Mark#setInteractionMode](#)

Example

```
// get
var interactionMode = mark.interactionMode;

// set
mark.interactionMode =
PCCViewer.Mark.InteractionMode.SelectionDisabled;
```

label :string

Gets or sets the text in the Stamp Mark.

This property is defined on marks of type: StampAnnotation, StampRedaction.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getLabel](#)
[PCCViewer.Mark#setLabel](#)

Example

```
if ('label' in mark) {
    var oldValue = mark.label;
    mark.label = "Approved";
}
```

mask :object

Gets or sets the mask for text input signatures.

This property is defined on marks of type: TextInputSignature.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- object

See: [PCCViewer.Mark#getMask](#)
[PCCViewer.Mark#setMask](#)

Example


```
if ('mask' in mark) {
    var oldValue = mark.mask;

    mark.mask = {
        value: '(###) ###-####',
        translations: {
            '#': /\d/
        }
    };
}
```

maxFontSize :number

Gets or sets a value that determines the maximum font size for a mark.

This method is defined on marks of type: `TextAreaSignature`.

This is an ECMA 5 property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.Mark#setMaxFontSize](#)
[PCCViewer.Mark#getMaxFontSize](#)

Example

```
// get
var maxFontSize = mark.maxFontSize;

// set
mark.maxFontSize = 13;
```

maxLength :number

Gets or sets a value that determines the max length of text for a mark

This method is defined on marks of type: `TextAnnotation`, `TextRedaction`, `TextAreaSignature` and `TextInputSignature`.

This is an ECMA 5 property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.Mark#setMaxLength](#)

PCCViewer.Mark#getMaxLength

Example

```
// get
var maxLength = mark.maxLength;

// set
mark.maxLength = 13;
```

opacity :number

Gets or sets the opacity of the Mark. This value is a number between 0 and 255.

This property is defined on marks of type: LineAnnotation, RectangleAnnotation, EllipseAnnotation, TextAnnotation, PolylineAnnotation.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.Mark#getOpacity](#)
[PCCViewer.Mark#setOpacity](#)

Example

```
if ('opacity' in mark) {
    var oldValue = mark.opacity;
    mark.opacity = 127; // set the opacity so that the
    mark is transparent
}
```

(readonly) pageNumber :number

Gets the page number of the page that the Mark is on.

This property is defined on all Mark objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.Mark#getPageNumber](#)

Example

```
var pageNumber = mark.pageNumber;
```

path :string

Gets or sets the path data of FreehandSignature and FreehandAnnotation.

This property is defined on marks of type: `FreehandSignature`, `FreehandAnnotation`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getPath](#)
[PCCViewer.Mark#setPath](#)

Example

```
if ('path' in mark) {  
    var oldValue = mark.path;  
    mark.path = "M0,0L1,1L1,0";  
}
```

position :Object

Gets or sets the position of the Mark.

Important - The setter only works if the ViewerControl instance has page text for all pages that the text-based mark will span. See [PCCViewer.Mark#setPosition](#) for complete information on safely setting the mark's position.

This property is defined on marks of type: `HighlightAnnotation`, `TextSelectionRedaction`, `TextHyperlinkAnnotation`, and `StrikethroughAnnotation`

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.Mark#getPosition](#)
[PCCViewer.Mark#setPosition](#)

Example

```
// Basic usage.
if ('position' in mark) {
    var oldValue = mark.position;

    // Requesting page text ensures that the
    ViewerControl has page text for the page (which
    // is a pre-requisite) and it also gives us the page
    text so that we have context when
    // setting the position.

viewerControl.requestPageText(1).then(function(pageText)
{
    // It is unsafe to set the mark position without
    first
    mark.position = {startIndex: 0, length:
pageText.length};
    });
}
```

reason :string

Gets or sets the reason in the Mark.

This property is defined on marks of type: `RectangleRedaction`, `TextSelectionRedaction`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Note: Throws an error if the `ViewerControlOptions.enableMultipleRedactionReasons` property was set to `true` when initializing `ViewerControl`. In such case you should use the `PCCViewer.Mark#reasons` instead.

Type:

- string

See: [PCCViewer.Mark#getReason](#)
[PCCViewer.Mark#setReason](#)

Example

```
if ('reason' in mark) {
    var oldValue = mark.reason;
    mark.reason = "Information for top security clearance
only.";
}
```

reasons :Array.<string>

Gets or sets the reasons in the Mark.

This property is defined on marks of type: `RectangleRedaction`, `TextSelectionRedaction`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Note: Throws an error if the `ViewerControlOptions.enableMultipleRedactionReasons` property was set to `false` when initializing `ViewerControl`. In such case you should use the `PCCViewer.Mark#reason` instead.

Type:

- Array.<string>

See: [PCCViewer.Mark#getReasons](#)
[PCCViewer.Mark#setReasons](#)

Example

```
if ('reasons' in mark) {
    var oldValue = mark.reasons;
    mark.reasons = ["1.a", "2.b", "Private Information"];
}
```

rectangle :Object

Gets or sets the bounding rectangle of the Mark.

This property is defined on marks of type: `RectangleAnnotation`, `EllipseAnnotation`, `TextAnnotation`, `StampAnnotation`, `RectangleRedaction`, `TransparentRectangleRedaction`, `TextRedaction`, `StampRedaction`, `FreehandAnnotation`, `FreehandSignature`, `TextSignature`, `ImageStampAnnotation`, `ImageStampRedaction`, `TextInputSignature` and `TextAreaSignature`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.Mark#getRectangle](#)
[PCCViewer.Mark#setRectangle](#)

Example

```
if ('rectangle' in mark) {
    var oldValue = mark.rectangle;
    mark.rectangle = {x: 0, y: 0, width: 100, height:
100};
}
```

signature :string

Note: This property is defined on the template mark of the `PlaceSignature` mouse tool, and is not available on any mark.

Gets or sets the template signature.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getSignature](#)
[PCCViewer.Mark#setSignature](#)

startPoint :Object

Gets or sets the start point coordinates of the line Mark.

This property is defined on marks of type: `LineAnnotation`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.Mark#getStartPoint](#)
[PCCViewer.Mark#setStartPoint](#)

Example

```
if ('startPoint' in mark) {
    var oldValue = mark.startPoint;
    mark.startPoint = {x: 1, y: 1}; // set the start
point to (1, 1)
}
```

text :string

Gets or sets the text in the Mark.

This property is defined on marks of type: `TextAnnotation`, `TextRedaction`, `TextSignature`, `TextAreaSignature`, `HighlightAnnotation`, `TextSelectionRedaction`, `TextHyperlinkAnnotation` and `StrikethroughAnnotation`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getText](#)
[PCCViewer.Mark#setText](#)

Example

```
if ('text' in mark) {
    var oldValue = mark.text;
    mark.text = "This is a Test";
}
```

thickness :number

Gets or sets the thickness of the Mark.

This property is defined on marks of type: `LineAnnotation`, `FreehandAnnotation`, `FreehandSignature`, `PolylineAnnotation`.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- number

See: [PCCViewer.Mark#getThickness](#)
[PCCViewer.Mark#setThickness](#)

Example

```
if ('thickness' in mark) {
    var oldValue = mark.thickness;
    mark.thickness = 3; // set the thickness of the mark
}
```

(readonly) **type** :string

Gets the type of the Mark.

This property is defined on all Mark objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark.Type](#)
[PCCViewer.Mark#getType](#)

Example

```
switch (mark.type) {
    case Mark.Type.LineAnnotation:
        ...
        break;
    default:
        ...
}
```

visible :string

Gets or sets the Mark visible.

This property is defined on all mark types.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Mark#getVisible](#)
[PCCViewer.Mark#setVisible](#)

Example

```
if ('visible' in mark) {
    var oldValue = mark.visible;
    mark.visible = true; // set mark to visible
}
```


Methods

`clearHighlights()` → `{PCCViewer.Mark}`

Clears any text highlights within the mark object. The text highlights would have been created with `PCCViewer.Mark#highlightText`.

It is invalid to call this method on the template mark of a `PCCViewer.MouseTool` object.

This method is defined on marks of type: `TextAnnotation`, `TextRedaction`, `HighlightAnnotation`, `TextSelectionRedaction`, `TextHyperlinkAnnotation`, and `StrikethroughAnnotation`.

See: [PCCViewer.Mark#highlightText](#)

Throws:

- If the mark is the template mark of a `PCCViewer.MouseTool` object.

Type
Error

- If the mark was created with the constructor `new Mark()`.

Type
Error

Returns:

The Mark object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
if (mark.clearHighlights) {
    mark.clearHighlights();
}
```

`getBorderColor()` → `{string}`

Gets the border color of the Mark.

This method is defined on marks of type: `RectangleAnnotation`, `EllipseAnnotation`, `TextAnnotation` and `RectangleRedaction`.

See: [PCCViewer.Mark#setBorderColor](#)

[PCCViewer.Mark#borderColor](#)**Returns:**

The border color of the Mark as a hexadecimal string.

Type
string

Example

```
if (mark.getBorderColor) {  
    var borderColor = mark.getBorderColor();  
}
```

[getBorderThickness\(\)](#) → {number}

Gets the border thickness of the mark.

This method is defined on marks of type: `RectangleAnnotation`, `EllipseAnnotation`, `TextAnnotation` and `RectangleRedaction`.

See: [PCCViewer.Mark#setBorderThickness](#)
[PCCViewer.Mark#borderThickness](#)

Returns:

The border thickness of the mark.

Type
number

Example

```
if (mark.getBorderThickness) {  
    var borderThickness = mark.getBorderThickness();  
}
```

[getBoundingRectangle\(\)](#) → {Object}

Gets the bounding rectangle for the Mark.

This method is defined on all Mark objects.

See: [PCCViewer.Mark#boundingRectangle](#)

Returns:

A rectangle object of the type {x: xValue, y: yValue, width: widthValue, height: heightValue}.

Type
Object

Example

```
var boundingRectangle = mark.getBoundingRectangle();
```

getColor() → {string}

Gets the color of the Mark.

This method is defined on marks of type: LineAnnotation, StampAnnotation, FreehandAnnotation, FreehandSignature, TextSignature, PolylineAnnotation and StrikethroughAnnotation.

See: [PCCViewer.Mark#setColor](#)
[PCCViewer.Mark#color](#)

Returns:

The color of the Mark as a hexadecimal string.

Type
string

Example

```
if (mark.getColor) {  
    var color = mark.getColor();  
}
```

getConversation() → {PCCViewer.Conversation}

Gets the conversation associated with the Mark.

This method is defined on all Mark objects.

Returns:

The conversation associated with this Mark.

Type
[PCCViewer.Conversation](#)

`getData(key)` → {string|object}

Gets the data value for the given key, or gets a hash containing all key values, if a key was not provided.

This method is defined on all Mark objects.

Parameters:

Name	Type	Description
key	string	The key for which to get the data value.

See: [PCCViewer.Data#getData](#)
[PCCViewer.Mark#setData](#)
[PCCViewer.Mark#getDataKeys](#)

Throws:

If the `key` argument is null or otherwise not a string.

Type
Error

Returns:

- If a key argument was provided, it returns the associated value.
- If a key argument was provided, but a value has not been associated with the key, then it returns undefined.
- If a key was not provided, it returns a hash object containing all key-value pairs.

Type
string | object

Example

```
var mark = viewerControl.addMark(1,
  "RectangleRedaction");

// The key "Author" is set the value "Mark".
mark.setData("Author", "Mark");

// The key "Note" is set the value "This is really
important!".
mark.setData("Note", "This is really important!");

mark.getData("Author"); // returns "Mark"
mark.getData();        // returns {"Author":"Mark",
```

```
"Note":"This is really important!"}
mark.getData("FooBar"); // returns undefined
```

`getDataKeys()` → {Array.<string>}

Gets an array of data keys known to this Mark.

This method is defined on all Mark objects.

See: [PCCViewer.Data#getDataKeys](#)
[PCCViewer.Mark#getData](#)
[PCCViewer.Mark#setData](#)

Returns:

Returns an array of data keys known to this Mark. If no data is stored, then an empty array will be returned.

Type

Array.<string>

Example

```
var mark = viewerControl.addMark(1,
"RectangleRedaction");

// Returns an empty array before key-value pairs are
stored.
mark.getDataKeys(); // returns []

// Returns a list of all keys.
mark.setData("Author", "Mark");
mark.setData("Note", "This is really important!");
mark.getDataKeys(); // returns ["Author", "Note"]
```

`getEndHeadType()` → {string|PCCViewer.Mark.LineHeadType}

Gets the line Mark end head type.

This method is defined on marks of type: LineAnnotation.

See: [PCCViewer.Mark#setEndHeadType](#)
[PCCViewer.Mark#endHeadType](#)

Returns:

A line head type enum value.

Type

string | [PCCViewer.Mark.LineHeadType](#)

Example

```
if (mark.getEndHeadType) {  
    var endHeadType = mark.getEndHeadType();  
}
```

`getEndPoint()` → `{Object}`

Gets the end point coordinates of the line Mark.

This method is defined on marks of type: `LineAnnotation`.

See: [PCCViewer.Mark#setEndPoint](#)
[PCCViewer.Mark#endPoint](#)

Returns:

A point object of the type `{x: xvalue, y: yvalue}`

Type

Object

Example

```
if (mark.getEndPoint) {  
    var endPoint = mark.getEndPoint();  
}
```

`getFillColor()` → `{string}`

Gets the fill color of the Mark.

This method is defined on marks of type: `RectangleAnnotation`, `EllipseAnnotation`, `TextAnnotation`, `HighlightAnnotation`, `'TextHyperlinkAnnotation'` and `RectangleRedaction`.

See: [PCCViewer.Mark#setFillColor](#)
[PCCViewer.Mark#fillColor](#)

Returns:

The fill color of the Mark as a hexadecimal string.

Type

string

Example

```
if (mark.getFillColor) {  
    var fillColor = mark.getFillColor();  
}
```

`getFontColor()` → {string}

Gets the font color of the text contained in the Text Mark.

This method is defined on marks of type: `TextAnnotation`, `TextRedaction`, `TextInputSignature`, `RectangleRedaction` and `TextAreaSignature`.

See: [PCCViewer.Mark#setFontColor](#)
[PCCViewer.Mark#fontColor](#)

Returns:

The text contained in the Text mark.

Type
string

Example

```
if (mark.getFontColor) {  
    var text = mark.getFontColor();  
}
```

`getFontName()` → {string}

Gets the font color of the text contained in the Mark.

This method is defined on marks of type: `TextAnnotation`, `TextRedaction`, `TextSignature`, `TextInputSignature` and `TextAreaSignature`.

See: [PCCViewer.Mark#setFontName](#)
[PCCViewer.Mark#fontName](#)

Returns:

The text contained in the Text mark.

Type
string

Example

```
if (mark.getFontName) {  
    var text = mark.getFontName();  
}
```

getFontSize() → {number}

Gets the font size (in points) of the text in the Mark.

This method is defined on marks of type: `TextAnnotation`, `TextRedaction`.

See: [PCCViewer.Mark#setFontSize](#)
[PCCViewer.Mark#fontSize](#)

Returns:

The font size of the text in the text mark.

Type
number

Example

```
if (mark.getFontSize) {  
    var fontSize = mark.getFontSize();  
}
```

getFontStyle() → {Array.<string>}

Gets an array of font styles of the text contained in the mark.

This method is defined on marks of type: `TextAnnotation`, `TextAreaSignature`, `TextRedaction`.

See: [PCCViewer.Mark.FontStyles](#)
[PCCViewer.Mark#setFontStyle](#)
[PCCViewer.Mark#fontStyle](#)

Returns:

An array containing the font styles of text contained in the Text mark.

Type
Array.<string>

Example

```
if (mark.getFontStyle) {  
    var fontStyleArray = mark.getFontStyle();  
}
```

`getHorizontalAlignment()` → {string}

Gets the horizontalAlignment of the text contained in the Text Mark.

This method is defined on marks of type: `TextAnnotation`, `TextRedaction`, `FreehandSignature`, `TextSignature`, `TextInputSignature` and `TextAreaSignature`.

See: [PCCViewer.Mark.HorizontalAlignment](#)
[PCCViewer.Mark#setHorizontalAlignment](#)
[PCCViewer.Mark#horizontalAlignment](#)

Returns:

A string containing horizontalAlignment contained in the Text mark.

Type
string

Example

```
if (mark.getHorizontalAlignment) {  
    var horizontalAlignment =  
    mark.getHorizontalAlignment();  
}
```

`getHref()` → {string|number}

Gets the link target for hyperlink annotations.

This method is defined on marks of type: `TextHyperlinkAnnotation`.

See: [PCCViewer.Mark#setHref](#)
[PCCViewer.Mark#href](#)

Returns:

The link target.

Type
string | number

Example

```
if (mark.getHref) {
    var href = mark.getHref();

    switch (typeof href) {
        case "number":
            // navigate to the page
            viewerControl.setPageNumber(href);
            break;
        case "string":
            // Interpret the URL and execute the
navigation.
            window.location.href = href;
            break;
        case "undefined":
        case "object":
        default:
            // do nothing, or define some special rules
            break;
    }
}
```

getId() → {string}

Gets the ID of the Mark.

This method is defined on all Mark objects.

See: [PCCViewer.Mark#id](#)

Returns:

The ID of the Mark.

Type
string

Example

```
var markId = mark.getId();
```

getImage() → {PCCViewer.Mark~ImageData}

Gets the image that is displayed for the Mark.

This method is defined on marks of type: ImageStampAnnotation and ImageStampRedaction.

See: [PCCViewer.Mark#setImage](#)

PCCViewer.Mark#image

Returns:

An object that represents the image to be shown for the mark.

Type

[PCCViewer.Mark~ImageData](#)

Example

```
if (mark.getImage) {  
    var imageData = mark.getImage();  
}
```

[getInteractionMode\(\)](#) → {string}

Gets a value that indicates the allowed interactions with this mark.

This method is defined on all [Mark](#) objects.

See: [PCCViewer.Mark#setInteractionMode](#)
[PCCViewer.Mark#interactionMode](#)

Returns:

A string value from the enumeration [PCCViewer.Mark.InteractionMode](#), which indicates the allowed interactions with this mark.

Type

string

Example

```
var interactionMode = mark.getInteractionMode();
```

[getLabel\(\)](#) → {string}

Gets the text string contained in the Stamp Mark.

This method is defined on marks of type: [StampAnnotation](#), [StampRedaction](#).

See: [PCCViewer.Mark#setLabel](#)
[PCCViewer.Mark#label](#)

Returns:

The text string in the Stamp mark.

Type
string

Example

```
if (mark.getLabel) {  
    var label = mark.getLabel();  
}
```

`getMask()` → {object}

Gets the applied mask for the text input signature mark.

This method is defined on marks of type: `TextInputSignature`.

See: [PCCViewer.Mark#setMask](#)
[PCCViewer.Mark#mask](#)

Returns:

The mask for the mark.

Type
object

Example

```
if (mark.getMask) {  
    var mask = mark.getMask();  
}
```

`getMaxFontSize()` → {Number}

Gets the maximum font size (in points) for text in the mark.

This method is defined on marks of type: `TextAreaSignature`.

See: [PCCViewer.Mark#setMaxFontSize](#)
[PCCViewer.Mark#maxFontSize](#)

Returns:

The maximum font size of the mark.

Type
Number

Example

```
if (mark.getMaxFontSize) {  
    var mask = mark.getMaxFontSize();  
}
```

`getMaxLength()` → {Number}

Gets the applied max length for the mark.

This method is defined on marks of type: `TextAnnotation`, `TextRedaction`, `TextAreaSignature` and `TextInputSignature`.

See: [PCCViewer.Mark#setMaxLength](#)
[PCCViewer.Mark#maxLength](#)

Returns:

The max length for the mark.

Type
Number

Example

```
if (mark.getMaxLength) {  
    var mask = mark.getMaxLength();  
}
```

`getOpacity()` → {number}

Gets the opacity of the Mark. This value is a number between 0 and 255.

This method is defined on marks of type: `LineAnnotation`, `RectangleAnnotation`, `EllipseAnnotation`, `TextAnnotation`.

See: [PCCViewer.Mark#setOpacity](#)
[PCCViewer.Mark#opacity](#)

Returns:

The opacity of the line.

Type

number

Example

```
if (mark.getOpacity) {  
    var opacity = mark.getOpacity();  
}
```

`getPageNumber()` → {number}

Gets the page number where the Mark object is located.

This method is defined on all Mark objects.

See: [PCCViewer.Mark#pageNumber](#)

Returns:

The page number where the Mark is located.

Type

number

Example

```
var pageNumber = mark.getPageNumber();
```

`getPath()` → {string}

Gets the path data for FreehandSignature and FreehandAnnotation.

This method is defined on marks of type: `FreehandSignature`, `FreehandAnnotation`.

See: [PCCViewer.Mark#setPath](#)
[PCCViewer.Mark#path](#)

Returns:

The path data string.

Type

string

Example

```
if (mark.getPath) {
```

```
    var path = mark.getPath();  
}
```

getPoints() → {Array}

Gets the array of points that make up coordinates of the vertices of the PolylineAnnotation Mark.

This method is defined on marks of type: PolylineAnnotation.

See: [PCCViewer.Mark#setPoints](#)
[PCCViewer.Mark#points](#)

Returns:

of point objects of the type {x: xvalue, y: yvalue}

Type
Array

Example

```
if (mark.getPoints) {  
    var points = mark.getPoints();  
}
```

getPosition() → {Object}

Gets the position of the text-based Mark.

This method is defined on marks of type: HighlightAnnotation, TextSelectionRedaction, TextHyperlinkAnnotation, and StrikethroughAnnotation.

See: [PCCViewer.Mark#setPosition](#)
[PCCViewer.Mark#position](#)

Returns:

A position object of the type {startIndex: startIndexValue, length: lengthValue}.

Type
Object

Example

```
if (mark.getPosition) {  
    var position = mark.getPosition();  
}
```

```
getReason() → {string}
```

Gets the reason contained in the Redaction Mark.

This method is defined on marks of type: `RectangleRedaction`, `TextSelectionRedaction`.

See: [PCCViewer.Mark#setReason](#)
[PCCViewer.Mark#reason](#)

Throws:

If the `ViewerControlOptions.enableMultipleRedactionReasons` property was set to `true` when initializing `ViewerControl`. In such case you should use the [PCCViewer.Mark#getReasons](#) instead.

Type
Error

Returns:

The reason contained in the Redaction mark.

Type
string

Example

```
if (mark.getReason) {  
    var reason = mark.getReason();  
}
```

```
getReasons() → {Array.<string>}
```

Gets the reasons contained in the Redaction Mark.

This method is defined on marks of type: `RectangleRedaction`, `TextSelectionRedaction`.

See: [PCCViewer.Mark#setReasons](#)
[PCCViewer.Mark#reasons](#)

Throws:

If the `ViewerControlOptions.enableMultipleRedactionReasons` property was set to `false` when initializing `ViewerControl`. In such case you should use the [PCCViewer.Mark#getReason](#) instead.

Type
Error

Returns:

The reasons contained in the Redaction mark.

Type
Array.<string>

Example

```
if (mark.getReasons) {  
    var reasons = mark.getReasons();  
}
```

getRectangle() → {Object}

Gets the bounding rectangle for the Mark.

This method is defined on marks of type: `RectangleAnnotation`, `EllipseAnnotation`, `TextAnnotation`, `StampAnnotation`, `RectangleRedaction`, `TransparentRectangleRedaction`, `TextRedaction`, `StampRedaction`, `FreehandAnnotation`, `FreehandSignature`, `TextSignature`, `ImageStampAnnotation`, `ImageStampRedaction`, `TextInputSignature` and `TextAreaSignature`.

See: [PCCViewer.Mark#setRectangle](#)
[PCCViewer.Mark#rectangle](#)

Returns:

An object of the type `{x: xValue, y: yValue, width: widthValue, height: heightValue}` describing the rectangle in pixels.

Type
Object

Example

```
if (mark.getRectangle) {  
    var boundingRectangle = mark.getRectangle();  
}
```

getSessionData(key) → {string|object}

Gets the session data value for the given key, or gets a hash containing all key values, if a key was not

provided. Unlike [PCCViewer.Mark#getData](#), this data is not saved with the annotation, it only lasts for the session.

This method is defined on all Mark objects.

Parameters:

Name	Type	Description
key	string	The key for which to get the data value.

See: [PCCViewer.Data#getSessionData](#)
[PCCViewer.Mark#setSessionData](#)
[PCCViewer.Mark#getSessionDataKeys](#)

Throws:

If the `key` argument is null or otherwise not a string.

Type
Error

Returns:

- If a key argument was provided, it returns the associated value.
- If a key argument was provided, but a value has not been associated with the key, then it returns undefined.
- If a key was not provided, it returns a hash object containing all key-value pairs.

Type
string | object

Example

```
var mark = viewerControl.addMark(1,
  "RectangleRedaction");

// The key "Visibility" is set the value "Shown".
mark.setSessionData("Visibility", "Shown");

// The key "Note" is set the value "This is not going to
be saved!".
mark.setSessionData("Note", "This is not going to be
saved!");

mark.getSessionData("Visibility"); // returns "Shown"
mark.getSessionData();           // returns
{"Visibility":"Shown", "Note":"This is not going to be
saved!"}
```

```
mark.getSessionData("FooBar"); // returns undefined
```

`getSessionDataKeys()` → {Array.<string>}

Gets an array of data keys known to this Mark. Unlike [PCCViewer.Mark#getDataKeys](#), this data is not saved with the annotation, it only lasts for the session.

This method is defined on all Mark objects.

See: [PCCViewer.Data#getSessionDataKeys](#)
[PCCViewer.Mark#getSessionData](#)
[PCCViewer.Mark#setSessionData](#)

Returns:

Returns an array of data keys known to this Mark. If no data is stored, then an empty array will be returned.

Type
Array.<string>

Example

```
var mark = viewerControl.addMark(1,
"RectangleRedaction");

// Returns an empty array before key-value pairs are
stored.
mark.getSessionDataKeys(); // returns []

// Returns a list of all keys.
mark.setSessionData("Visibility", "Shown");
mark.setSessionData("Note", "This is not going to be
saved!");
mark.getSessionDataKeys(); // returns ["Visibility",
"Note"]
```

`getSignature()` → {Object|undefined}

Note: This property is defined on the template mark of the `PlaceSignature` mouse tool, and is not available on any mark.

Gets the last signature object that was associated with the particular template mark.

See: [PCCViewer.Mark#setSignature](#)
[PCCViewer.Mark#signature](#)

Returns:

The `PlaceSignature` object, or undefined. See [PCCViewer.Signatures~FreehandSignature](#) and [PCCViewer.Signatures~TextSignature](#).

Type

Object | undefined

Example

```
// get the mouse tool
var accusoftPlaceSignature =
PCCViewer.MouseTools.getMouseTool('AccusoftPlaceSignature');

// get the template mark
var signatureTemplateMark =
accusoftPlaceSignature.getTemplateMark();
// get the signature associated with the template
var signature = signatureTemplateMark.getSignature();
```

`getStartPoint()` → {Object}

Gets the start point coordinates of the line Mark.

This method is defined on marks of type: `LineAnnotation`.

See: [PCCViewer.Mark#setStartPoint](#)
[PCCViewer.Mark#startPoint](#)

Returns:

A point object of the type `{x: xvalue, y: yvalue}`

Type

Object

Example

```
if (mark.getStartPoint) {
    var startPoint = mark.getStartPoint();
}
```

`getText()` → {string}

Gets the text contained in marks with text or text-selection based marks.

This method is defined on marks of type: `TextAnnotation`, `TextRedaction`, `TextSignature`, `HighlightAnnotation`, `TextSelectionRedaction`, `TextHyperlinkAnnotation`,

TextInputSignature, StrikethroughAnnotation and TextAreaSignature.

See: [PCCViewer.Mark#setText](#)
[PCCViewer.Mark#text](#)

Returns:

The text contained in the in the above mentioned mark types.

Type
string

Example

```
if (mark.getText) {  
    var text = mark.getText();  
}
```

getThickness() → {number}

Gets the thickness of the line.

This method is defined on marks of type: LineAnnotation, FreehandAnnotation, FreehandSignature, PolylineAnnotation, StrikethroughAnnotation.

See: [PCCViewer.Mark#setThickness](#)
[PCCViewer.Mark#thickness](#)

Returns:

The thickness of the line.

Type
number

Example

```
if (mark.getThickness) {  
    var thickness = mark.getThickness();  
}
```

getType() → {string}

Gets the type of the Mark type.

This method is defined on all Mark objects.

See: [PCCViewer.Mark.Type](#) for a list of possible Mark types.
[PCCViewer.Mark.Type](#)
[PCCViewer.Mark#type](#)

Returns:

The type of Mark.

Type
string

Example

```
switch (mark.getType()) {  
    case Mark.Type.LineAnnotation:  
        ...  
        break;  
    default:  
        ...  
}
```

[setVisible\(\)](#) → {boolean}

Gets the visibility of the Mark.

This method is defined on all mark types.

See: [PCCViewer.Mark#setVisible](#)
[PCCViewer.Mark#visible](#)

Returns:

Returns true if the mark is visible, false otherwise.

Type
boolean

Example

```
if (mark.getVisible) {  
    var visible = mark.getVisible();  
}
```

[highlightText\(highlights\)](#) → {PCCViewer.Mark}

Highlights text within the mark. The highlights are not persisted when mark is saved using saveMarkup.

Existing highlights that were created with a previous call to this method, are cleared when this method is called.

It is invalid to call this method on the template mark of a [PCCViewer.MouseTool](#) object.

This method is defined on marks of type: `TextAnnotation`, `TextRedaction`, `HighlightAnnotation`, `TextSelectionRedaction`, `TextHyperlinkAnnotation` and `StrikethroughAnnotation`.

Parameters:

Name	Type	Description
highlights	Array. <object> object	<p>An array of objects or a single object that defines a highlight.</p> <ul style="list-style-type: none"> • Each object has the following properties: <ul style="list-style-type: none"> ◦ <code>startIndex</code> {number} - required <ul style="list-style-type: none"> ▪ The start index of the highlight in the mark text. ▪ The valid range is $0 \leq \text{startIndex} < \text{mark.getText().length}$. ◦ <code>length</code> {number} - required <ul style="list-style-type: none"> ▪ The length of the highlight in the mark. ▪ If the length is greater than the number of remaining characters in the mark, then the remaining characters in the mark will be highlighted. The excessive length value will be ignored. ▪ The valid range is $\text{length} > 0$. ◦ <code>color</code> {string} - required <ul style="list-style-type: none"> ▪ Specifies the Hexadecimal color for the highlight. ▪ Valid values are any 7-character string representing a color. The first letter must be a "#" symbol and the other six characters must be hexadecimal digits representing the red, green, and blue portions of the color. ◦ <code>opacity</code> {number} - optional <ul style="list-style-type: none"> ▪ Specifies the opacity of the highlight. ▪ Valid values are from 0 to 255 (inclusive). ▪ If unspecified, an opacity value of 127 will be used. <p>If passed a value of null, undefined, or an empty array, then the highlights are cleared.</p>

See: [PCCViewer.Mark#clearHighlights](#)
[PCCViewer.Mark#getText](#)

Throws:

- If any of the highlights in `highlights` param are missing required properties or contain invalid data.

Type
Error

- If the mark is the template mark of a [PCCViewer.MouseTool](#) object.

Type
Error

- If the mark was created with the constructor `new Mark()`.

Type
Error

Returns:

The Mark object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
if (mark.highlightText) {
    mark.highlightText([
        {startIndex: 0, length: 5, color: "#FF0000"},
        {startIndex: 10, length: 5, color: "#FF0000"},
    ], opacity: 200);
}
```

[setBorderColor\(value\)](#) → {[PCCViewer.Mark](#)}

Sets the border color of the Mark.

This method is defined on marks of type: `RectangleAnnotation`, `EllipseAnnotation`, `TextAnnotation` and `RectangleRedaction`.

Parameters:

Name	Type	Description
value	string	Hexadecimal string representing border color. This string must be prepended with '#' character.

See: [PCCViewer.Mark#getBorderColor](#)
[PCCViewer.Mark#borderColor](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setBorderColor) {
    mark.setBorderColor("#FF0000"); // set the border
    color to red
}
```

setBorderThickness(value) → [{PCCViewer.Mark}](#)

Sets the border thickness of the mark.

This method is defined on marks of type: [RectangleAnnotation](#), [EllipseAnnotation](#), [TextAnnotation](#) and [RectangleRedaction](#).

Parameters:

Name	Type	Description
value	number	Border thickness of the mark.

See: [PCCViewer.Mark#getBorderThickness](#)
[PCCViewer.Mark#borderThickness](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setBorderThickness) {
    mark.setBorderThickness(3);
}
```

setColor(value) → [{PCCViewer.Mark}](#)

Sets the color of the Mark.

This method is defined on marks of type: [LineAnnotation](#), [StampAnnotation](#), [FreehandAnnotation](#), [FreehandSignature](#), [TextSignature](#), [PolylineAnnotation](#) and [StrikethroughAnnotation](#).

Parameters:

Name	Type	Description
value	string	Hexadecimal string representing color. This string must be prepended with '#' character.

See: [PCCViewer.Mark#getColor](#)
[PCCViewer.Mark#color](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setColor) {
    mark.setColor("#FF0000"); // set the mark's color to
    red
}
```

setData(key, value) → {PCCViewer.Mark}

Sets the data value for the given key.

This method is defined on all Mark objects.

Notes:

- Overwrites any data value already associated with the given key.
- There is no artificial limit imposed on the number of key-value pairs that are stored.
- If limits on the number of key-value pairs are required, they should be enforced by calling code.
- Setting the value as undefined results in no information for the key being persisted to the server.
- The returned data is not mutated or sanitized, which could lead to a security vulnerability if not sanitized properly before use.

Parameters:

Name	Type	Description
key	string	The key for which to set the data value.
value	string	This is the value to set for the key. <ul style="list-style-type: none"> • This must be a string or undefined. • The maximum length of the string is not limited by this function.

See: [PCCViewer.Data#setData](#)
[PCCViewer.Mark#getData](#)
[PCCViewer.Mark#getDataKeys](#)

Returns:

The Mark object on which the method was called.

Type

[PCCViewer.Mark](#)

Example

```
var mark = viewerControl.addMark(1,
  "RectangleRedaction");

// Get data returns undefined before the key is set.
mark.getData("Author"); // returns undefined

// The key "Author" is set the value "Mark".
mark.setData("Author", "Mark");
mark.getData("Author"); // returns "Mark"

// The key "Author" is overwritten with the value
"Clark".
mark.setData("Author", "Clark");
mark.getData("Author"); // returns "Clark"

// The key "Author" is unset, by setting the value to
undefined.
mark.setData("Author", undefined);
mark.getData("Author"); // returns undefined

// The value can only be set to a string or undefined.
// All other data types throw.
mark.setData("FooBar", null); // throws
mark.setData("FooBar", 1);    // throws
mark.setData("FooBar", true); // throws
mark.setData("FooBar", {});   // throws
mark.setData("FooBar", []);   // throws
```

setEndHeadType(value) → {[PCCViewer.Mark](#)}

Sets the line head type.

This method is defined on marks of type: `LineAnnotation`.

Parameters:

Name	Type	Description
value	PCCViewer.Mark.LineHeadType	The line head type. For example, <code>PCCViewer.Mark.LineHeadType.FilledRectangle</code> can be specified to make the line appear as an arrow.

See: [PCCViewer.Mark.LineHeadType](#)
[PCCViewer.Mark#getEndHeadType](#)
[PCCViewer.Mark#endHeadType](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setEndHeadType) {
    // Put a triangle head on the end of the line to make
    it an arrow
    mark.setEndHeadType("FilledTriangle");
    // or use the enumeration to accomplish the same
    thing

    mark.setEndHeadType(PCCViewer.Mark.LineHeadType.FilledTriang
}
}
```

setEndPoint(value) → {[PCCViewer.Mark](#)}

Sets the end point coordinate of the line Mark.

This method is defined on marks of type: `LineAnnotation`.

Parameters:

Name	Type	Description
value	Object	Start point coordinates of a line Mark. The parameter object must be of the following type: {x: xvalue, y: yvalue}

See: [PCCViewer.Mark#getEndPoint](#)
[PCCViewer.Mark#endPoint](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setEndPoint) {
    mark.setEndPoint({x:100, y:100});
}
```

setFillColor(value) → {PCCViewer.Mark}

Sets the fill color of the Mark.

This method is defined on marks of type: `RectangleAnnotation`, `EllipseAnnotation`, `TextAnnotation`, `HighlightAnnotation`, `'TextHyperlinkAnnotation'` and `RectangleRedaction`.

Parameters:

Name	Type	Description
value	string	Hexadecimal string representing fill color. This string must be prepended with '#' character.

See: [PCCViewer.Mark#getFillColor](#)
[PCCViewer.Mark#fillColor](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setFillColor) {
    mark.setFillColor("#FF0000"); // set the fill color
    to red
}
```

setFontColor(value) → {PCCViewer.Mark}

Sets the font color of the text in the Text Mark.

This method is defined on marks of type: `TextAnnotation`, `TextRedaction`, `TextInputSignature`, `RectangleRedaction` and `TextAreaSignature`.

Parameters:

Name	Type	Description
value	string	A string value containing the hexadecimal color for the text of the text annotation.

See: [PCCViewer.Mark#getFontColor](#)
[PCCViewer.Mark#fontColor](#)

Returns:

The Mark object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
if (mark.setFontColor) {  
    mark.setFontColor("#000000");  
}
```

setFontName(value) → {[PCCViewer.Mark](#)}

Sets the font name of the text in the Text Mark.

This method is defined on marks of type: `TextAnnotation`, `TextRedaction`, `TextSignature`, `TextInputSignature` and `TextAreaSignature`.

Parameters:

Name	Type	Description
value	string	A string value containing the font name for the text in the text annotation.

See: [PCCViewer.Mark#getFontName](#)
[PCCViewer.Mark#fontName](#)

Returns:

The Mark object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
if (mark.setFontName) {  
    mark.setFontName("Aerial");  
}
```

setFontSize(value) → {PCCViewer.Mark}

Sets the font size (in points) for the text to use in the Mark.

Note: The API uses the resolution of the image to determine the size of a point so, for example, a line of 12 point text on a 300 DPI raster image will be 12 points / 72 point-per-inch * 300 pixels-per-inch = 50 pixels tall. The default value is 12 points.

This method is defined on marks of type: `TextAnnotation`, `TextRedaction`.

Parameters:

Name	Type	Description
value	number	A number for the font size for the text in the text annotation.

See: [PCCViewer.Mark#getFontSize](#)
[PCCViewer.Mark#fontSize](#)

Returns:

The Mark object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
if (mark.setFontSize) {  
    mark.setFontSize(12);  
}
```

setFontStyle(value) → {PCCViewer.Mark}

Sets the font styles provided in the parameter array of the text in the mark.

This method is defined on marks of type: `TextAnnotation`, `TextAreaSignature`, `TextRedaction`.

Parameters:

Name	Type	Description
value	Array. <string>	<p>An array containing values containing the font styles for the text in the text annotation.</p> <p>Valid values in the array are:</p> <ul style="list-style-type: none"> • "Bold" (PCCViewer.Mark.FontStyles.Bold) • "Italic" (PCCViewer.Mark.FontStyles.Italic) • "Underline" (PCCViewer.Mark.FontStyles.Underline) • "Strikeout" (PCCViewer.Mark.FontStyles.Strikeout) <p>Note: An empty array would render the text with normal font style.</p>

See: [PCCViewer.Mark.FontStyles](#)
[PCCViewer.Mark#getFontStyle](#)
[PCCViewer.Mark#fontStyle](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setFontStyle) {
    var fontStylesArray = ["Bold", "Italic", "Underline"];
    mark.setFontStyle(fontStylesArray);
}
```

[setHorizontalAlignment\(value\)](#) → {[PCCViewer.Mark](#)}

Sets the horizontal alignment of the text in the Text Mark.

This method is defined on marks of type: [TextAnnotation](#), [TextRedaction](#), [FreehandSignature](#), [TextSignature](#), [TextInputSignature](#) and [TextAreaSignature](#).

Parameters:

Name	Type	Description
value	string	A string value containing the horizontal alignment for the text in the text annotation.

See: [PCCViewer.Mark.HorizontalAlignment](#)
[PCCViewer.Mark#getHorizontalAlignment](#)
[PCCViewer.Mark#horizontalAlignment](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setHorizontalAlignment) {
    mark.setHorizontalAlignment("center");
}
```

setHref(href) → {[PCCViewer.Mark](#)}

Sets the link target for hyperlink annotations.

All strings and numbers are valid values. It is the responsibility of the API consumer to handle clicks of hyperlink annotations. When handling the click, the API consumer should interpret the href value and take the appropriate navigation action.

This method is defined on marks of type: `TextHyperlinkAnnotation`.

Parameters:

Name	Type	Description
href	string number	The link target.

See: [PCCViewer.Mark#getHref](#)
[PCCViewer.Mark#href](#)
[PCCViewer.EventType.Click](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setHref) {
    // set to fully qualified URL
    mark.setHref("http://www.accusoft.com/");

    // or a URL fragment
    mark.setHref("#named-annotation");
}
```

```
// or a numeric value
mark.setHref(4);
}
```

setImage(imageData) → {[PCCViewer.Mark](#)}

Sets the image that is displayed for the Mark.

This method is defined on marks of type: [ImageStampAnnotation](#) and [ImageStampRedaction](#).

Parameters:

Name	Type	Description
imageData	PCCViewer.Mark~ImageData	An object that represents the image to be shown for the mark.

See: [PCCViewer.Mark#getImage](#)
[PCCViewer.Mark#image](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
var param = {
  dataUrl: "data:image/png;base64,base64 string",
  id: "imageId"
};
if (mark.setImage) {
  mark.setImage(param);
}
```

setInteractionMode(interactionMode) → {[PCCViewer.ViewerControl](#)}

Sets a value that indicates the allowed interactions with this mark.

This method is defined on all [Mark](#) objects.

Parameters:

Name	Type	Description
interactionMode	string	A string value from the enumeration PCCViewer.Mark.InteractionMode , which indicates the allowed interactions with this mark.

See: [PCCViewer.Mark#getInteractionMode](#)
[PCCViewer.Mark#interactionMode](#)

Returns:

The object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
mark.setInteractionMode(PCCViewer.Mark.InteractionMode.Full)
```

setLabel(value) → {PCCViewer.Mark}

Sets the text string in the Stamp Mark.

This method is defined on marks of type: StampAnnotation, StampRedaction.

Parameters:

Name	Type	Description
value	string	A string value containing the text in the Stamp annotation.

See: [PCCViewer.Mark#getLabel](#)
[PCCViewer.Mark#label](#)

Returns:

The Mark object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
if (mark.setLabel) {  
    mark.setLabel("Approved");  
}
```

setMask(mask) → {PCCViewer.Mark}

Sets the mask for the text input signature mark. Once this mark enters edit mode, the user input will be

masked according to the properties set using this method.

This method is defined on marks of type: `TextInputSignature`.

Parameters:

Name	Type	Description									
mask	object	The mask to set on this mark to assist user input. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>value</td><td>string</td><td>The string representation of the mask. The user input will <i>look</i> like this string once they have finished their input. Each character in this string that does not have a translation will be represented to the user literally.</td></tr><tr><td>translations</td><td>object</td><td>The translations to use for the given mask value. The key represents a character present in the mask value, and the value is a regular expression which validates the acceptable user input for that character.</td></tr></tbody></table>	Name	Type	Description	value	string	The string representation of the mask. The user input will <i>look</i> like this string once they have finished their input. Each character in this string that does not have a translation will be represented to the user literally.	translations	object	The translations to use for the given mask value. The key represents a character present in the mask value, and the value is a regular expression which validates the acceptable user input for that character.
Name	Type	Description									
value	string	The string representation of the mask. The user input will <i>look</i> like this string once they have finished their input. Each character in this string that does not have a translation will be represented to the user literally.									
translations	object	The translations to use for the given mask value. The key represents a character present in the mask value, and the value is a regular expression which validates the acceptable user input for that character.									

See: [PCCViewer.Mark#getMask](#)
[PCCViewer.Mark#mask](#)
[PCCViewer.ViewerControl#enterTextMarkEditMode](#)

Throws:

- If `mask` is not an object, undefined or null.
Type
Error
- If `mask.value` is not a string.
Type
Error
- If `mask.translations` is not an object.
Type
Error
- If `mask.translations` contains a key with a string length not equal to 1.
Type

Error

- If `mask.translations` contains a value that is not a regular expression.

Type

Error

- If `mask.translations` contains a value that is a regular expression with flags.

Type

Error

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)**Example**

```
if (mark.setMask) {
  // only allow a US phone number as input
  mark.setMask({
    value: '(###) ###-####'
    translations: {
      '#': /\d/
    }
  });

  // only allow an Arizona driver's license number
  mark.setMask({
    value: 'A#####-AA#####-#####-A#####',
    translations: {
      'A': /[a-zA-Z]/,
      '#': /\d/
    }
  });
}
```

setMaxFontSize(maxFontSize) → [{PCCViewer.Mark}](#)

Sets the maximum font size (in points) for text in the mark. Setting a value of 0 indicates no max font size (in this case, the text will enlarge to fit to the mark bounds no matter how large the mark is).

This method is defined on marks of type: `TextAreaSignature`.

Parameters:

Name	Type	Description
maxFontSize	number	The positive number to set as the maximum font size of the mark.

See: [PCCViewer.Mark#getMaxFontSize](#)
[PCCViewer.Mark#maxFontSize](#)

Throws:

If `maxFontSize` is not a positive integer or 0.

Type
 Error

Returns:

The Mark object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
if (mark.setMaxFontSize) {
    // only allow the font to enlarge to 72px
    mark.setMaxFontSize(72);
}
```

`setMaxLength(maxLength)` → `{PCCViewer.Mark}`

Sets the maximum number of characters that may be entered into an input

This method is defined on marks of type: `TextAnnotation`, `TextRedaction`, `TextAreaSignature`, and `TextInputSignature`.

Parameters:

Name	Type	Description
maxLength	number	The positive number to set as the max length of the mark

See: [PCCViewer.Mark#getMaxLength](#)
[PCCViewer.Mark#maxLength](#)

Throws:

If `maxLength` is not a positive integer or 0.

Type
Error

Returns:

The Mark object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
if (mark.setMaxLength) {  
    // only allow 5 characters to be entered  
    mark.setMaxLength(5);  
}
```

`setOpacity(value)` → [{PCCViewer.Mark}](#)

Sets the opacity of the Mark. This value is a number between 0 and 255.

This method is defined on marks of type: `LineAnnotation`, `RectangleAnnotation`, `EllipseAnnotation`, `TextAnnotation`, `FreehandAnnotation`, `PolylineAnnotation`.

Parameters:

Name	Type	Description
<code>value</code>	number	Opacity of the Mark. Acceptable values are in the range 0 to 255.

See: [PCCViewer.Mark#getOpacity](#)
[PCCViewer.Mark#opacity](#)

Returns:

The Mark object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
if (mark.setOpacity) {  
    mark.setOpacity(255); // fully opaque  
    mark.setOpacity(127); // semi-transparent  
    mark.setOpacity(0); // transparent  
}
```

```
}
```

setPath(path) → {PCCViewer.Mark}

Sets the path data of FreehandSignature and FreehandAnnotation.

This method is defined on marks of type: FreehandSignature, FreehandAnnotation.

Parameters:

Name	Type	Description
path	string	The path data string. This includes a subset of the SVG path standard, including the M, L, and C commands only.

See: [PCCViewer.Mark#getPath](#)
[PCCViewer.Mark#path](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setPath) {  
    mark.setPath("M0,0L1,1L1,0");  
}
```

setPoints(value) → {PCCViewer.Mark}

Sets the points vertices coordinate array of the PolylineAnnotation Mark.

This method is defined on marks of type: PolylineAnnotation.

Parameters:

Name	Type	Description
value	Object	is an array of coordinates of the vertices in a Polyline Mark. Each point must be of the following type: {x: xvalue, y: yvalue}

See: [PCCViewer.Mark#getPoints](#)

Throws:

If `value` is not an array.

Type

Error

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setPoints) {
    mark.setPoints([{x:100, y:100}, {x:200, y: 250}...,
{x:1000, y: 1000});
}
```

[setPosition\(value\)](#) → [{PCCViewer.Mark}](#)

Sets the position of text-based Marks.

Important - This method only works if the ViewerControl instance has page text for all pages that the text-based mark will span. In common use cases the ViewerControl will have text for these pages, however the ViewerControl API provides methods to check if it has text for a page and also to force it to get text for a page.

There are certain methods that force the ViewerControl to get text for a page or all pages.

- Calling [PCCViewer.ViewerControl#requestPageText](#) will force the ViewerControl to get text for a specified page if it does not already have the text for the page. This method also has the benefit of providing you with the page text, so that you are not blindly setting the position of the mark on the page.
- Calling [PCCViewer.ViewerControl#search](#) will force the viewer to get text for all pages by the time the search completes.

There are means to determine if the ViewerControl has text for a page.

- Calling [PCCViewer.ViewerControl#isPageTextReady](#) will synchronously indicate if the viewer has text for a page.
- The ViewerControl will trigger the [PCCViewer.EventType.PageTextReady](#) event when it gets text for a page.

This method is defined on marks of type: `HighlightAnnotation`, `TextSelectionRedaction`, `TextHyperlinkAnnotation`, and `StrikethroughAnnotation`.

Parameters:

Name	Type	Description
value	Object	Position of a Mark. The parameter object must be of the following type: {startIndex: startIndexValue, length: lengthValue}

See: [PCCViewer.Mark#getPosition](#)
[PCCViewer.Mark#position](#)

Throws:

- If the viewer does not have text for any page that the mark will span.
Type
Error
- If `position.startIndex` is not a valid number that indicates the index of a character on the mark's page.
Type
Error
- If `position.length` is negative or will cause the mark to extend past the last character in the document.
Type
Error
- If `position` does not contain the property `startIndex` or `length`.
Type
Error

Returns:

The Mark object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
// Example 1
// -----
// Basic (unsafe) usage, call setPosition and pass in an
// object with a startIndex and length.
if (mark.setPosition) {
```

```
        mark.setPosition({startIndex:0, length:5});
    }

    // Example 2
    // -----
    // Safe usage, request the page text and then highlight
    // something within that page.
    // This code highlights the half of the characters on the
    // page.
    viewerControl.requestPageText(1).then(
        function(pageText) {
            // Now that we have the page text for page 1,
            // Add a HighlightAnnotation to page 1 and set
            // the position of the highlight.
            viewerControl.addMark(1,
                PCCViewer.Mark.Type.HighlightAnnotation)
                .setPosition({startIndex: pageText.length /
                    4, length: pageText.length / 2});
        });

    // Example 3
    // -----
    // Safe highlighting of arbitrary spans of text.
    // We highlight 3000 characters starting at index 100 on
    // page 2.
    var markPage = 2,
        markPosition = {startIndex: 100, length: 3000};

    // Since the highlight will be 3000 characters, it may
    // span multiple pages.
    // We use a helper method to ensure that the
    // ViewerControl has text for all pages that it will span.
    ensureViewerControlHasPageText(viewerControl, markPage,
        markPosition)
        .then(addHighlightAnnotation,
            function(error) {
                alert("Something went wrong when trying to
                    get page text. " + (error.message ? error.message :
                    error));
            });

    // This function uses the ViewerControl API to add a
    // HighlightAnnotation.
    // It will be called when ensureViewerControlHasPageText
    // completes.
    function addHighlightAnnotation() {
        // Add the HighlightAnnotation using
        // ViewerControl#addMark and then
        // set the position and color of the highlight.
        viewerControl.addMark(markPage,
            PCCViewer.Mark.Type.HighlightAnnotation)
```

```
        .setPosition(markPosition)
        .setFillColor("#FF0000");

    // Scroll to the page containing the mark.
    viewerControl.setPageNumber(markPage);
}

// A helper method to ensure that a ViewerControl
instance has page text
// for all pages that a HighlightAnnotation or
TextSelectionRedaction spans.
function ensureViewerControlHasPageText(viewerControl,
markPageNumber, markPostion) {

    // Calling ViewerControl#requestPageText will cause
the ViewerControl to get text
    // from the server if it does not already have it.
This method also gives us the page
    // text so we can check if the mark will extend to
the next page.
    return
viewerControl.requestPageText(markPageNumber).then(

        // If requestPageText promise is fulfilled, we
compare the markPosition to the
        // page text, and if necessary, recursively
ensure text for the next page.
        function(pageText) {
            // Check for an invalid markPosition. The
method setPosition will now allow the caller
            // to crate a mark that starts after the last
character on the page.
            if (markPostion.startIndex >=
pageText.length) {
                throw new Error("Mark cannot start after
last character on the page.");
            }

            // Determine if the highlight extends into
the next page.
            var remainingCharsOnPage = pageText.length -
markPostion.startIndex;
            var remainingCharsInHighlight =
markPostion.length - remainingCharsOnPage;
            var extendsToNextPage =
remainingCharsInHighlight > 0;

            // If the highlight extends to the next page,
and we are not on the last page,
            // then ensure the viewer has the text for
the next page.
```

```
        if (extendsToNextPage) {
            if (markPageNumber <
viewerControl.getPageCount()) {
                return
ensureViewerControlHasPageText(viewerControl,
markPageNumber + 1,
                                {startIndex: 0, length:
remainingCharsInHighlight});
            }
            // Mark#setPosition does not allow the
mark to extend off of the document.
            else {
                throw new Error("Mark cannot extend
off the document.")
            }
        }
    }
}
)
```

setReason(value) → {[PCCViewer.Mark](#)}

Sets the reason in the Redaction Mark.

This method is defined on marks of type: [RectangleRedaction](#), [TextSelectionRedaction](#).

Parameters:

Name	Type	Description
value	string	Redaction reason of the Mark. Acceptable values are any string.

See: [PCCViewer.Mark#getReason](#)
[PCCViewer.Mark#reason](#)

Throws:

If the [ViewerControlOptions.enableMultipleRedactionReasons](#) property was set to true when initializing [ViewerControl](#). In such case you should use the [PCCViewer.Mark#setReasons](#) instead.

Type
Error

Returns:

The Mark object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
if (mark.setReason) {
    mark.setReason("Information for top security
clearance only.");
}
```

setReasons(value) → {[PCCViewer.Mark](#)}

Sets the reasons in the Redaction Mark.

This method is defined on marks of type: [RectangleRedaction](#), [TextSelectionRedaction](#).

Parameters:

Name	Type	Description
value	Array.<string>	Redaction reasons of the Mark. Acceptable values are an array of any string.

See: [PCCViewer.Mark#getReasons](#)
[PCCViewer.Mark#reasons](#)

Throws:

If the [ViewerControlOptions.enableMultipleRedactionReasons](#) property was set to false when initializing [ViewerControl](#). In such case you should use the [PCCViewer.Mark#setReason](#) instead.

Type
Error

Returns:

The Mark object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
if (mark.setReasons) {
    mark.setReasons(["1.a", "2.b", "Private
Information"]);
}
```

setRectangle(value) → {[PCCViewer.Mark](#)}

Sets the bounding rectangle of the Mark.

This method is defined on marks of type: `RectangleAnnotation`, `EllipseAnnotation`, `TextAnnotation`, `StampAnnotation`, `RectangleRedaction`, `TransparentRectangleRedaction`, `TextRedaction`, `StampRedaction`, `FreehandAnnotation`, `FreehandSignature`, `TextSignature`, `ImageStampAnnotation`, `ImageStampRedaction`, `TextInputSignature` and `TextAreaSignature`.

Parameters:

Name	Type	Description
value	Object	Bounding rectangle of a Mark. The parameter object must be of the following type: <code>{x: xValue, y: yValue, width: widthValue, height: heightValue}</code>

See: [PCCViewer.Mark#getRectangle](#)
[PCCViewer.Mark#rectangle](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setRectangle) {
    mark.setRectangle({x:0, y:0, width:100, height:100});
}
```

`setSessionData(key, value)` → [{PCCViewer.Mark}](#)

Sets the session data value for the given key. Unlike [PCCViewer.Mark#setData](#), this data is not saved with the annotation, it only lasts for the session.

This method is defined on all Mark objects.

Parameters:

Name	Type	Description
key	string	The key for which to set the data value.
value	string	This is the value to set for the key. <ul style="list-style-type: none"> This must be a string or undefined. The maximum length of the string is not limited by this function.

See: [PCCViewer.Data#setSessionData](#)
[PCCViewer.Mark#getSessionData](#)
[PCCViewer.Mark#getSessionDataKeys](#)

Returns:

The Mark object on which the method was called.

Type

[PCCViewer.Mark](#)

Example

```
var mark = viewerControl.addMark(1,
  "RectangleRedaction");

// Get data returns undefined before the key is set.
mark.getSessionData("Visibility"); // returns undefined

// The key "Visibility" is set the value "Shown".
mark.setSessionData("Visibility", "Shown");
mark.getSessionData("Visibility"); // returns "Shown"

// The key "Visibility" is overwritten with the value
"Hidden".
mark.setSessionData("Visibility", "Hidden");
mark.getSessionData("Visibility"); // returns "Hidden"

// The key "Visibility" is unset, by setting the value to
undefined.
mark.setSessionData("Visibility", undefined);
mark.getSessionData("Visibility"); // returns undefined

// The value can only be set to a string or undefined.
// All other data types throw.
mark.setSessionData("FooBar", null); // throws
mark.setSessionData("FooBar", 1); // throws
mark.setSessionData("FooBar", true); // throws
mark.setSessionData("FooBar", {}); // throws
mark.setSessionData("FooBar", []); // throws
```

setSignature(signature) → {[PCCViewer.Mark](#)}

Note: This property is defined on the template mark of the PlaceSignature mouse tool, and is not available on any mark.

Sets the signature data to use by the PlaceSignature mouse tool.

Parameters:

Name	Type	Description
signature	Object undefined	An object with properties that specify the signature data. Using <code>undefined</code> will reset the state of the mouse tool back to default. See PCCViewer.Signatures~FreehandSignature and PCCViewer.Signatures~TextSignature

See: [PCCViewer.Mark#getSignature](#)
[PCCViewer.Mark#signature](#)

Throws:

- If `signature` is not an `Object` or `undefined`.

Type
Error

- If the signature object does not have either a `path` or `text` string property.

Type
Error

Returns:

The `Mark` object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
// get the mouse tool
var accusoftPlaceSignature =
PCCViewer.MouseTools.getMouseTool('AccusoftPlaceSignature');

// get the template mark
var signatureTemplateMark =
accusoftPlaceSignature.getTemplateMark();

// set signature path for freehand signature
signatureTemplateMark.setSignature({path:
"M0,0L100,0L100,100L0,100L0,0"});

// the same template can be reused for text signature
signatureTemplateMark.setSignature({ text: "Joe Smith",
```

```
fontName: "Arial" });
```

setStartPoint(value) → {PCCViewer.Mark}

Sets the start point coordinate of the line Mark.

This method is defined on marks of type: LineAnnotation.

Parameters:

Name	Type	Description
value	Object	Start point coordinates of a line Mark. The parameter object must be of the following type: {x:xvalue, y: yvalue}

See: [PCCViewer.Mark#getStartPoint](#)
[PCCViewer.Mark#startPoint](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setStartPoint) {  
    mark.setStartPoint({x:1, y:1});  
}
```

setText(value) → {PCCViewer.Mark}

Sets the text in the Text Mark.

This method is defined on marks of type: TextAnnotation, TextRedaction, TextSignature, TextInputSignature and TextAreaSignature.

Note: This method is NOT available for marks of type: HighlightAnnotation, TextSelectionRedaction, TextHyperlinkAnnotation and StrikethroughAnnotation

Parameters:

Name	Type	Description
value	string	Text of the Mark. Acceptable values are any string.

See: [PCCViewer.Mark#getText](#)
[PCCViewer.Mark#text](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setText) {
    mark.setText("This is test Text");
}
```

[setThickness\(value\)](#) → [{PCCViewer.Mark}](#)

Sets the thickness of line.

This method is defined on marks of type: [LineAnnotation](#), [FreehandAnnotation](#), [FreehandSignature](#), [PolylineAnnotation](#), [StrikethroughAnnotation](#).

Parameters:

Name	Type	Description
value	number	Thickness of the line.

See: [PCCViewer.Mark#getThickness](#)
[PCCViewer.Mark#thickness](#)

Returns:

The Mark object on which this method was called.

Type

[PCCViewer.Mark](#)

Example

```
if (mark.setThickness) {
    mark.setThickness(3);
}
```

[setVisible\(\)](#) → [{PCCViewer.Mark}](#)

Sets the Mark to either visible or invisible depending on the boolean parameter.

This method is defined on all mark types.

Parameters:

Name	Type	Description
value.	boolean	

See: [PCCViewer.Mark#getVisible](#)
[PCCViewer.Mark#visible](#)

Returns:

The Mark object on which this method was called.

Type
[PCCViewer.Mark](#)

Example

```
if (mark.setVisible) {
    mark.setVisible(true); // sets the mark visible
}
```

Type Definitions

ImageData

This type is used to specify the image data and a unique identifier for the image data. Objects of this type are used by the [PCCViewer.Mark#getImage](#), [PCCViewer.Mark#setImage](#), and [PCCViewer.Mark#image](#) members.

Type:

- Object

Properties:

Name	Description
<code>id</code> : string	An arbitrary string id used to identify this image. Though this can be any string, it is expected that the same string identifier be used to identify the same image data.
<code>dataUrl</code> : string	A base64-encoded data URL representation of an image.

Class: MarkupLayer

PCCViewer. MarkupLayer

(protected) `new MarkupLayer(viewerControl, markReferencesopt)`

The `MarkupLayer` object is used to group together marks and their associated comments. A layer may be persisted to the web tier using `PCCViewer.ViewerControl#saveMarkupLayer`. Also, a layer may be stored in a `PCCViewer.MarkupLayerCollection` where it can be retrieved for later use.

When creating a layer, mark references may be added. A mark reference is a JSON object that represents a comment that refers to a mark on another layer. When the current user comments on a mark that does not exist in his layer, then his persisted layer record will contain a reference to the mark while the full mark will exist in another record. Because records might be loaded out of order or in an incomplete set, this parameter provides a way to store the mark reference. If the record containing the full mark loads later then the data stored here can be attached to it.

After creating a layer, marks may be added and removed from it.

The `MarkupLayer` object also provides an event subscription method, to get notified of other types of information. See `PCCViewer.MarkupLayer.EventType`.

Parameters:

Name	Type	Attributes	Description
<code>viewerControl</code>	<code>string</code>		The <code>PCCViewer.ViewerControl</code> for the loaded document.
<code>markReferences</code>	<code>Object Array. <Object></code>	<code><optional></code>	The JSON reference node (or an array of them) from the markup layer record.

Example

```
// Optionally, specify any references to marks on another
layer or layers
var markReference = {
  creationDateTime: "2015-06-12T21:20:58.527Z",
  data: {
    "key1": "value1",
    "key2": "value2"
  },
  markUid:
  "ZZZrOV8yMDE1LTA2LTExVDE5OjU5OjEwLjE3MlplfNDg2dzI5",
  text: "user 1 comment on user 3 mark"
};

// Create a new layer
var layer = new PCCViewer.MarkupLayer(viewerControl,
```

```
markReference);

//register some events
layer
    .on(PCCViewer.MarkupLayer.EventType.MarkupLayerCreated,
function(ev) {
    alert("Markup layer created.");
})

    .on(PCCViewer.MarkupLayer.EventType.MarkupLayerDestroyed,
function(ev) {
    alert("Markup layer destroyed.");
})
    .on(PCCViewer.MarkupLayer.EventType.MarksAddedToLayer,
function(ev) {
    alert("Mark added to layer: " +
ev.marks[0].getId());
})

    .on(PCCViewer.MarkupLayer.EventType.MarksRemovedFromLayer,
function(ev) {
    alert("Mark removed from layer: " +
ev.marks[0].getId());
})

    .on(PCCViewer.MarkupLayer.EventType.MarkupLayerInteractionMode
function(ev) {
    alert("Layer's interaction mode changed to: " +
ev.interactionMode);
})
    .on(PCCViewer.MarkupLayer.EventType.MarkupLayerHidden,
function(ev) {
    alert("MarkupLayer#hide() called.");
})
    .on(PCCViewer.MarkupLayer.EventType.MarkupLayerShown,
function(ev) {
    alert("MarkupLayer#show() called.");
});

// Create a new mark
var mark = viewerControl.addMark(1, "HighlightAnnotation");
// Add the mark to the layer
layer.addMarks(mark);
// Determine if a mark is contained in a layer
var markInLayer = layer.hasMark(mark.getId());
```

Members

(static, readonly) **EventType** :string

A list of events that can be triggered by the [PCCViewer.MarkupLayer](#) object.

Note: This enumeration is for convenience for API developers. Instead of using this enumeration, you can pass string values of the eventType (enumeration values)

Type:

- string

Properties:

Name	Description
MarkupLayerCreated : string	
MarkupLayerDestroyed : string	
MarksAddedToLayer : string	
MarksRemovedFromLayer : string	
MarkupLayerInteractionModeChanged : string	
MarkupLayerHidden : string	
MarkupLayerShown : string	

See: [PCCViewer.Event](#)
[PCCViewer.MarkupLayer#on](#)
[PCCViewer.MarkupLayer#off](#)

(readonly) id :Object

Gets the layer's ID.

This property is defined on all MarkupLayer objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.MarkupLayer#getId](#)

Example

```
var id = MarkupLayer.id;
```

name :Object

Gets and sets the layer's name.

This property is defined on all MarkupLayer objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.MarkupLayer#getName](#)
[PCCViewer.MarkupLayer#setName](#)

Example

```
var name = MarkupLayer.name;
```

originalXmlName :Object

Gets and sets the layer's original XML name.

This property is defined on all MarkupLayer objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.MarkupLayer#getOriginalXmlName](#)
[PCCViewer.MarkupLayer#setOriginalXmlName](#)

Example

```
var name = MarkupLayer.originalXmlName;
```

(readonly) recordId :Object

Gets the ID of web tier record from which this layer was created.

This property is defined on all MarkupLayer objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.MarkupLayer#getRecordId](#)

Example

```
var recordId = MarkupLayer.recordId;
```

(readonly) **viewerControl** :Object

Gets the viewer control associated with this layer.

This property is defined on all MarkupLayer objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.MarkupLayer#getViewerControl](#)

Example

```
var viewerControl = MarkupLayer.viewerControl;
```

Methods

addMarks(A) → {[PCCViewer.MarkupLayer](#)}

Used to add marks to the layer.

Parameters:

Name	Type	Description
A	PCCViewer.Mark Array.< PCCViewer.Mark >	{ PCCViewer.Mark } object or an array of them.

Returns:

The markup layer Object.

Type

[PCCViewer.MarkupLayer](#)

Example

```
var markupLayer =
viewerControl.getMarkupLayerCollection().getAll()[0]; //
Get the first markup layer.
var mark = viewerControl.addMark(1, 'HighlightAnnotation');
// Create a new mark
markupLayer.addMarks(mark);
```

copyLayers(marks) → {PCCViewer.MarkupLayer}

Copies marks from other layers to this layer.

Note: This method requires that attributes of each page referenced by the layer marks have been obtained by the viewer prior to calling. Use [PCCViewer.ViewerControl#requestPageAttributes](#) to obtain the necessary page attributes before calling this method.

Note: The copied marks are assigned new unique IDs, and any references to the original mark (such as a comment on the mark that is stored in another layer) will not reference the copied mark. A copy of each comment on the mark is put on the copy of the mark in this layer.

Parameters:

Name	Type	Description
marks	Array.< PCCViewer.MarkupLayer >	An array of markup layers to copy to this layer.

Throws:

If markupLayers is not an array of markup layers known to the viewer.

Type
Error

Returns:

The markup layer Object on which this method was called.

Type
[PCCViewer.MarkupLayer](#)

Example

```
var markupLayer1 =
viewerControl.getMarkupLayerCollection().getAll()[0]; //
Get the first markup layer.
var markupLayer2 =
viewerControl.getMarkupLayerCollection().getAll()[1]; //
Get the second markup layer.
var markupLayer4 =
```

```
viewerControl.getMarkupLayerCollection().getAll()[3]; //
Get the fourth markup layer.
// Concatenate layers 2 and 4 to a single array.
var layersToCopy = markupLayer2.concat(markupLayer4);
// Copy the layers to this layer.
markupLayer1.copyLayers(layersToCopy);
```

destroy()

This method will remove the layer from the viewer control's markup layer collection. Also, it will de-register any event listeners associated with the layer.

Example

```
var markupLayer =
viewerControl.getMarkupLayerCollection().getAll()[0]; //
Get the first markup layer.
markupLayer.destroy();
```

getData(key) → {string|object}

Gets the data value for the given key, or gets a hash containing all key values, if a key was not provided. This method is defined on all MarkupLayer objects.

Parameters:

Name	Type	Description
key	string	The key for which to get the data value.

See: [PCCViewer.Data#getData](#)
[PCCViewer.MarkupLayer#setData](#)
[PCCViewer.MarkupLayer#getDataKeys](#)

Throws:

If the `key` argument is null or otherwise not a string.

Type
Error

Returns:

- If a key argument was provided, it returns the associated value.
- If a key argument was provided, but a value has not been associated with the key, then it returns undefined.
- If a key was not provided, it returns a hash object containing all key-value pairs.

Type

string | object

Example

```
// The key "Username" is set the value "Admin".
layer.setData("Username", "Admin");

// The key "CreateDate" is set the value "1970-01-01".
layer.setData("CreateDate", "1970-01-01");

layer.getData("Username"); // returns "Admin"
layer.getData();           // returns {"Username":"Admin",
"CreateDate":"1970-01-01"}
layer.getData("FooBar"); // returns undefined
```

getDataKeys() → {Array.<string>}

Gets an array of data keys known to this MarkupLayer.

This method is defined on all MarkupLayer objects.

See: [PCCViewer.Data#getDataKeys](#)
[PCCViewer.MarkupLayer#getData](#)
[PCCViewer.MarkupLayer#setData](#)

Returns:

Returns an array of data keys known to this MarkupLayer. If no data is stored, then an empty array will be returned.

Type

Array.<string>

Example

```
// Returns an empty array before key-value pairs are
stored.
layer.getDataKeys(); // returns []

// Returns a list of all keys.
layer.setData("Username", "Admin");
layer.setData("CreateDate", "1970-01-01");
layer.getDataKeys(); // returns ["Username", "CreateDate"]
```

getId() → {string}

Gets the layer's ID.

Returns:

The ID of the layer.

Type
string

getMarkReferences() → {Array.<Object>}

Gets the mark references associated with this layer.

Returns:

An array of mark reference Objects.

Type
Array.<Object>

Example

```
var markupLayer =  
viewerControl.getMarkupLayerCollection().getAll()[0]; //  
Get the first markup layer.  
var markReferences = markupLayer.getMarkReferences();
```

getMarks() → {Array.<PCCViewer.Mark>}

Gets the marks associated with this layer.

Returns:

An array of {PCCViewer.Mark} Objects.

Type
Array.<PCCViewer.Mark>

Example

```
var markupLayer =  
viewerControl.getMarkupLayerCollection().getAll()[0]; //  
Get the first markup layer.  
var marks = markupLayer.getMarks();
```

getName() → {string}

Gets the layer's name.

Returns:

The name of the layer.

Type
string

Example

```
var markupLayer =  
viewerControl.getMarkupLayerCollection().getAll()[0]; //  
Get the first markup layer.  
var layerName = markupLayer.getName();
```

getOriginalXmlName() → {string}

Gets the name of the web tier XML record from which the marks of this layer were originally stored.

Returns:

The name of the web tier XML record from which the marks of this layer were originally stored.

Type
string

Example

```
var markupLayer =  
viewerControl.getMarkupLayerCollection().getAll()[0]; //  
Get the first markup layer.  
var originalXmlName = markupLayer.getOriginalXmlName();
```

getRecordId() → {string}

Gets the ID of web tier record from which this layer was created.

Returns:

The layer record ID

Type

string

Example

```
var markupLayer =  
viewerControl.getMarkupLayerCollection().getAll()[0]; //  
Get the first markup layer.  
var recordId = markupLayer.getRecordId();
```

getSessionData(key) → {string|object}

Gets the session data value for the given key, or gets a hash containing all key values, if a key was not provided. Unlike [PCCViewer.MarkupLayer#getData](#), this data is not saved with the annotation, it only lasts for the session.

This method is defined on all MarkupLayer objects.

Parameters:

Name	Type	Description
key	string	The key for which to get the data value.

See: [PCCViewer.SessionData#getSessionData](#)
[PCCViewer.MarkupLayer#setSessionData](#)
[PCCViewer.MarkupLayer#getSessionDataKeys](#)

Throws:

If the `key` argument is null or otherwise not a string.

Type
Error

Returns:

- If a key argument was provided, it returns the associated value.
- If a key argument was provided, but a value has not been associated with the key, then it returns undefined.
- If a key was not provided, it returns a hash object containing all key-value pairs.

Type
string | object

Example

```
// The key "Username" is set the value "Admin".
```

```
layer.setSessionData("Username", "Admin");

// The key "CreatedDate" is set the value "1970-01-01".
layer.setSessionData("CreatedDate", "1970-01-01");

layer.getSessionData("Username"); // returns "Admin"
layer.getSessionData();           // returns
{"Username":"Admin", "CreatedDate":"1970-01-01"}
layer.getSessionData("FooBar");   // returns undefined
```

getSessionDataKeys() → {Array.<string>}

Gets an array of data keys known to this MarkupLayer. Unlike [PCCViewer.MarkupLayer#getDataKeys](#), this data is not saved with the annotation, it only lasts for the session.

This method is defined on all MarkupLayer objects.

See: [PCCViewer.SessionData#getSessionDataKeys](#)
[PCCViewer.MarkupLayer#getSessionData](#)
[PCCViewer.MarkupLayer#setSessionData](#)

Returns:

Returns an array of data keys known to this MarkupLayer. If no data is stored, then an empty array will be returned.

Type
Array.<string>

Example

```
// Returns an empty array before key-value pairs are
stored.
layer.getSessionDataKeys(); // returns []

// Returns a list of all keys.
layer.setSessionData("Username", "Admin");
layer.setSessionData("CreatedDate", "1970-01-01");
layer.getSessionDataKeys(); // returns ["Username",
"CreatedDate"]
```

getViewerControl() → {PCCViewer.ViewerControl}

Gets the viewer control associated with this layer.

Returns:

A viewer control object.

Type

[PCCViewer.ViewerControl](#)

Example

```
var markupLayer =
viewerControl.getMarkupLayerCollection().getAll()[0]; //
Get the first markup layer.
var viewerControl = markupLayer.getViewerControl();
```

hasMark(A) → {boolean}

Used to query the layer to see if it contains a particular mark.

Parameters:

Name	Type	Description
A	string	mark's id.

Returns:

A true or false indication depending on whether the mark exists in the layer.

Type

boolean

Example

```
var markupLayer =
viewerControl.getMarkupLayerCollection().getAll()[0]; //
Get the first markup layer.
var mark = viewerControl.getAllMarks()[0];
var markExistsInLayer = markupLayer.hasMark(mark.getId());
```

hide() → {PCCViewer.MarkupLayer}

Hides all of the marks in the layer.

Returns:

The markup layer Object on which this method was called.called.

Type

[PCCViewer.MarkupLayer](#)

Example

```
var markupLayer =  
viewerControl.getMarkupLayerCollection().getAll()[0]; //  
Get the first markup layer.  
markupLayer.hide(); // Hide the marks in the layer.
```

off(eventType, handler) → {PCCViewer.MarkupLayer}

Unsubscribe an event handler from a specified event type.

Typically, an event is unsubscribed when you no longer want further notification of the event.

Parameters:

Name	Type	Description
eventType	string	A string specifying the event type. See PCCViewer.MarkupLayer.EventType for a list and description of all supported events.
handler	PCCViewer.Event~eventHandler	A function that was attached previously to the <code>ViewerControl</code> . Note: This must be the same function object previously passed to PCCViewer.MarkupLayer#on . It cannot be a different object that is functionally equivalent.

See: [PCCViewer.MarkupLayer#on](#)
[PCCViewer.ViewerControl#off](#) for more details on unsubscribing event handlers.

Returns:

The `MarkupLayer` object on which this method was called.

Type

[PCCViewer.MarkupLayer](#)

on(eventType, handler) → {PCCViewer.MarkupLayer}

Subscribe an event handler to an event of a specified type.

Parameters:

Name	Type	Description
eventType	string	A string that specifies the event type. This value is case-insensitive. See PCCViewer.MarkupLayer.EventType for a list and description of all supported events.
handler	PCCViewer.Event~eventHandler	A function that will be called whenever the event is triggered.

See: [PCCViewer.MarkupLayer#off](#)
[PCCViewer.ViewerControl#on](#) for more details on event subscription.

Returns:

The MarkupLayer object on which this method was called.

Type

[PCCViewer.MarkupLayer](#)

removeMarks(A) → {[PCCViewer.MarkupLayer](#)}

Used to remove marks from the layer.

Parameters:

Name	Type	Description
A	PCCViewer.Mark Array.< PCCViewer.Mark >	{ PCCViewer.Mark } object or an array of them.

Returns:

The markup layer Object.

Type

[PCCViewer.MarkupLayer](#)

Example

```
var markupLayer = viewerControl.getActiveMarkupLayer(); //
Get the active markup layer.
var mark = viewerControl.addMark(1, 'HighlightAnnotation');
// Create a new mark
markupLayer.removeMarks(mark); // remove the mark from the
active layer
```

setData(key, value) → {[PCCViewer.MarkupLayer](#)}

Sets the data value for the given key.

This method is defined on all MarkupLayer objects.

Notes:

- Overwrites any data value already associated with the given key.
- There is no artificial limit imposed on the number of key-value pairs that are stored.
- If limits on the number of key-value pairs are required, they should be enforced by calling code.
- Setting the value as undefined results in no information for the key being persisted to the server.
- The returned data is not mutated or sanitized, which could lead to a security vulnerability if not sanitized properly before use.

Parameters:

Name	Type	Description
key	string	The key for which to set the data value.
value	string	This is the value to set for the key. <ul style="list-style-type: none">• This must be a string or undefined.• The maximum length of the string is not limited by this function.

See: [PCCViewer.Data#setData](#)
[PCCViewer.MarkupLayer#getData](#)
[PCCViewer.MarkupLayer#getDataKeys](#)

Returns:

The MarkupLayer object on which the method was called.

Type

[PCCViewer.MarkupLayer](#)

Example

```
// Get data returns undefined before the key is set.
layer.getData("Username"); // returns undefined

// The key "Username" is set the value "Admin".
layer.setData("Username", "Admin");
layer.getData("Username"); // returns "Admin"

// The key "Username" is overwritten with the value
"Guest1".
layer.setData("Username", "Guest1");
layer.getData("Username"); // returns "Guest1"

// The key "Username" is unset, by setting the value to
undefined.
```

```

layer.setData("Username", undefined);
layer.getData("Username"); // returns undefined

// The value can only be set to a string or undefined.
// All other data types throw.
layer.setData("FooBar", null); // throws
layer.setData("FooBar", 1);    // throws
layer.setData("FooBar", true); // throws
layer.setData("FooBar", {});   // throws
layer.setData("FooBar", []);   // throws

```

setInteractionMode(interactionMode) → {[PCCViewer.MarkupLayer](#)}

Used to alter the interaction mode of all marks in a layer.

Parameters:

Name	Type	Description
interactionMode	string	A string value from the enumeration PCCViewer.Mark.InteractionMode ,

Returns:

The markup layer Object on which this method was called

Type

[PCCViewer.MarkupLayer](#)

Example

```

var markupLayer =
viewerControl.getMarkupLayerCollection().getAll()[0]; //
Get the first markup layer.
markupLayer.setInteractionMode(PCCViewer.Mark.InteractionMode

```

setName(name) → {[PCCViewer.MarkupLayer](#)}

Sets the layer's name.

Parameters:

Name	Type	Description
name	string	The name to apply to this layer.

Returns:

The markup layer Object on which this method was called.

Type

[PCCViewer.MarkupLayer](#)

Example

```
var markupLayer =
viewerControl.getMarkupLayerCollection().getAll()[0]; //
Get the first markup layer.
markupLayer.setName('Final Draft');
```

setOriginalXmlName(name) → [{PCCViewer.MarkupLayer}](#)

Sets the name of the web tier XML record from which the marks of this layer were originally stored. If the layer is not associated with an XML file, the property should be set to an empty string.

When the original XML name is set and a markup layer is saved, the original XML name is saved in the markup layer JSON. When the [PCCViewer.ViewerControl#requestMarkupLayerNames](#) method is called, the original XML name will be provided.

Parameters:

Name	Type	Description
name	string	The name of the web tier XML record from which the marks of this layer were originally stored.

Returns:

The markup layer Object on which this method was called.

Type

[PCCViewer.MarkupLayer](#)

Example

```
var markupLayer =
viewerControl.getMarkupLayerCollection().getAll()[0]; //
Get the first markup layer.
markupLayer.setOriginalXmlName('my marks');
```

setSessionData(key, value) → [{PCCViewer.MarkupLayer}](#)

Sets the session data value for the given key. Unlike [PCCViewer.MarkupLayer#setData](#), this data is not saved with the annotation, it only lasts for the session.

This method is defined on all MarkupLayer objects.

Parameters:

Name	Type	Description
key	string	The key for which to set the data value.
value	string	This is the value to set for the key. <ul style="list-style-type: none">• This must be a string or undefined.• The maximum length of the string is not limited by this function.

See: [PCCViewer.SessionData#setSessionData](#)
[PCCViewer.MarkupLayer#getSessionData](#)
[PCCViewer.MarkupLayer#getSessionDataKeys](#)

Returns:

The MarkupLayer object on which the method was called.

Type

[PCCViewer.MarkupLayer](#)

Example

```
// Get data returns undefined before the key is set.
layer.getSessionData("Username"); // returns undefined

// The key "Username" is set the value "Admin".
layer.setSessionData("Username", "Admin");
layer.getSessionData("Username"); // returns "Admin"

// The key "Username" is overwritten with the value
"Guest1".
layer.setSessionData("Username", "Guest1");
layer.getSessionData("Username"); // returns "Guest1"

// The key "Username" is unset, by setting the value to
undefined.
layer.setSessionData("Username", undefined);
layer.getSessionData("Username"); // returns undefined

// The value can only be set to a string or undefined.
// All other data types throw.
layer.setSessionData("FooBar", null); // throws
layer.setSessionData("FooBar", 1); // throws
layer.setSessionData("FooBar", true); // throws
layer.setSessionData("FooBar", {}); // throws
layer.setSessionData("FooBar", []); // throws
```

`show()` → `{PCCViewer.MarkupLayer}`

Shows all of the marks in the layer.

Returns:

The markup layer Object on which this method was called.

Type

`PCCViewer.MarkupLayer`

Example

```
var markupLayer =
viewerControl.getMarkupLayerCollection().getAll()[0]; //
Get the first markup layer.
markupLayer.show(); // Show the marks in the layer.
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: MarkupLayerCollection

PCCViewer. MarkupLayerCollection

(protected) `new MarkupLayerCollection(viewerControl)`

The `MarkupLayerCollection` object is used to hold and manage the markup layers as they are added, removed, shown and hidden. These actions will determine what marks and comments are displayed.

After creating a `MarkupLayerCollection`, `PCCViewer.MarkupLayer` objects may be added and removed from it. Additionally, mark references may also be added to it.

The `MarkupLayerCollection` object also provides an event subscription method, to get notified of other types of information. See [PCCViewer.MarkupLayerCollection.EventType](#).

Parameters:

Name	Type	Description
<code>viewerControl</code>	<code>string</code>	The PCCViewer.ViewerControl for the loaded document.

Example

```
// Get the `MarkupLayerCollection` associated with the
viewerControl
```



```
var layerCollection =
viewerControl.getMarkupLayerCollection()
// Get all the layers in a collection
var layers = layerCollection.getAll();
// Remove a layer from a collection
layerCollection.remove(layers[0].getId());

//register some events
layerCollection

.on(PCCViewer.MarkupLayerCollection.EventType.MarkupLayerAdded

    function(ev) {
        alert("Markup layer added to the collection with
id = " + ev.layerId);
    })

.on(PCCViewer.MarkupLayerCollection.EventType.MarkupLayerRemov

    function(ev) {
        alert("Markup layer removed from the collection
with id = " + ev.layerId);
    });
```

Members

(static, readonly) **EventType** :string

A list of events that can be triggered by the [PCCViewer.MarkupLayerCollection](#) object.

Note: This enumeration is for convenience for API developers. Instead of using this enumeration, you can pass string values of the eventType (enumeration values)

Type:

- string

Properties:

Name	Description
MarkupLayerAdded : string	
MarkupLayerRemoved : string	

See: [PCCViewer.Event](#)
[PCCViewer.MarkupLayerCollection#on](#)
[PCCViewer.MarkupLayerCollection#off](#)

(readonly) `viewerControl` :Object

Gets the number representing how many layers are in the collection.

This property is defined on all MarkupLayerCollection objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.MarkupLayerCollection#getCount](#)

Example

```
var layerCount = MarkupLayerCollection.count;
```

(readonly) `viewerControl` :Object

Gets the viewer control associated with this markup layer collection.

This property is defined on all MarkupLayerCollection objects.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- Object

See: [PCCViewer.MarkupLayerCollection#getViewerControl](#)

Example

```
var viewerControl = MarkupLayerCollection.viewerControl;
```

Methods

`addItem(layer)` → {[PCCViewer.MarkupLayerCollection](#)}

This method is used to add a layer to the collection.

Parameters:

Name	Type	Description
layer	PCCViewer.MarkupLayer	A markup layer object.

Name	Type	Description
------	------	-------------

Returns:

The markup layer collection object on which this method was called.

Type

[PCCViewer.MarkupLayerCollection](#)

Example

```
var layerCollection =
viewerControl.getMarkupLayerCollection(); // Get the markup
layer collection associated with the viewerControl
var layer = new PCCViewer.MarkupLayer(viewerControl);
layerCollection.addItem(layer);
```

forEach(iterator, thisArg_{opt}) → [{PCCViewer.MarkupLayerCollection}](#)

A method to iterate over all items in the collection. This method matches the spec for `Array.prototype.forEach`.

Parameters:

Name	Type	Attributes	Description
iterator	function PCCViewer.MarkupLayerCollection~iterator		The function to execute for each item in the collection.
thisArg	*	<optional>	The Object to be used as <code>this</code> for the iterator function.

Throws:

If the `iterator` parameter is not a function.

Type

`TypeError`

Returns:

The markup layer collection Object on which this method was called.

Type

[PCCViewer.MarkupLayerCollection](#)

`getAll()` → {Array.<PCCViewer.MarkupLayer>}

Gets all the layers from the collection.

Returns:

An array of markup layer objects.

Type

Array.<PCCViewer.MarkupLayer>

Example

```
var layerCollection =
viewerControl.getMarkupLayerCollection(); // Get the markup
layer collection associated with the viewerControl
var layers = layerCollection.getAll();
```

`getCount()` → {number}

Gets the number representing how many layers are in the collection.

Returns:

The number representing how many layers are in the collection.

Type

number

Example

```
var layerCollection =
viewerControl.getMarkupLayerCollection(); // Get the markup
layer collection associated with the viewerControl
var layerCount = layerCollection.getCount();
```

`getItem(layerId)` → {PCCViewer.MarkupLayer|undefined}

Gets a specific layer from the collection.

Parameters:

Name	Type	Description
layerId	string	A layer ID that corresponds to a layer object contained in the collection.

Returns:

A markup layer object or `undefined` if `layerId` does not correspond to a layer in the collection.

Type

[PCCViewer.MarkupLayer](#) | `undefined`

Example

```
var layerCollection =
viewerControl.getMarkupLayerCollection(); // Get the markup
layer collection associated with the viewerControl
var layers = layerCollection.getAll();
var layer = layerCollection.getItem(layers[0].getId());
```

`getViewerControl()` → **`{PCCViewer.ViewerControl}`**

Gets the viewer control associated with this layer.

Returns:

A viewer control object.

Type

[PCCViewer.ViewerControl](#)

Example

```
var layerCollection =
viewerControl.getMarkupLayerCollection(); // Get the markup
layer collection associated with the viewerControl
var viewerControl = layerCollection.getViewerControl();
```

`off(eventType, handler)` → **`{PCCViewer.MarkupLayerCollection}`**

Unsubscribe an event handler from a specified event type.

Typically, an event is unsubscribed when you no longer want further notification of the event.

Parameters:

Name	Type	Description
<code>eventType</code>	<code>string</code>	A string specifying the event type. See PCCViewer.MarkupLayerCollection.EventType for a list and description of all supported events.
<code>handler</code>	PCCViewer.Event~eventHandler	A function that was attached previously to the <code>ViewerControl</code> .

Name	Type	Description
		Note: This must be the same function object previously passed to PCCViewer.MarkupLayerCollection#on . It cannot be a different object that is functionally equivalent.

See: [PCCViewer.MarkupLayerCollection#on](#)
[PCCViewer.ViewerControl#off](#) for more details on unsubscribing event handlers.

Returns:

The `MarkupLayerCollection` object on which this method was called.

Type
[PCCViewer.MarkupLayerCollection](#)

on(eventType, handler) → {PCCViewer.MarkupLayerCollection}

Subscribe an event handler to an event of a specified type.

Parameters:

Name	Type	Description
eventType	string	A string that specifies the event type. This value is case-insensitive. See PCCViewer.MarkupLayerCollection.EventType for a list and description of all supported events.
handler	PCCViewer.Event~eventHandler	A function that will be called whenever the event is triggered.

See: [PCCViewer.MarkupLayerCollection#off](#)
[PCCViewer.ViewerControl#on](#) for more details on event subscription.

Returns:

The `MarkupLayerCollection` object on which this method was called.

Type
[PCCViewer.MarkupLayerCollection](#)

removeAll() → {PCCViewer.MarkupLayerCollection}

This method is used to remove all layers from the collection.

Returns:

The markup layer collection object on which this method was called.

Type

[PCCViewer.MarkupLayerCollection](#)

Example

```
var layerCollection =
viewerControl.getMarkupLayerCollection(); // Get the markup
layer collection associated with the viewerControl
layerCollection.removeAll();
```

removeItem(layerId) → [{PCCViewer.MarkupLayerCollection}](#)

This method is used to remove a layer from the collection.

Parameters:

Name	Type	Description
layerId	string	An ID of a layer in the collection.

Returns:

The markup layer collection object on which this method was called.

Type

[PCCViewer.MarkupLayerCollection](#)

Example

```
var layerCollection =
viewerControl.getMarkupLayerCollection(); // Get the markup
layer collection associated with the viewerControl
var layers = layerCollection.getAll();
layerCollection.removeItem(layers[0].getId());
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: MouseTool

PCCViewer. **MouseTool**

This object represents an instance of a mouse tool. Each mouse tool is given a name and type when it is created. The name is used as a handle to the mouse tool. The type defines the behavior and properties of the mouse tool.

A named mouse tool is a global tool that is available to all viewers. Each viewer instance can access the same `MouseTool` object by name. This permits one mouse tool to be used by two different viewer instances at the same time.

Constructor

`new MouseTool()`

The `MouseTool` constructor is for internal use only. The appropriate way to create and register a new named mouse tool is to use the method `PCCViewer.MouseTools.createMouseTool`.

See: [PCCViewer.MouseTools.createMouseTool](#)

Throws:

If the type is unknown. See [PCCViewer.MouseTool.Type](#) for a list of known tool types.

Type
 RangeError

Example

```
// use the PCCViewer.MouseTools.createMouseTool(name, type)
// function instead of this constructor.
PCCViewer.MouseTools.createMouseTool("myMouseTool",
"LineAnnotation");
```

Members

(static, readonly) `Type` :string

This enumerable contains a list of all known tool types. There are used to create new `PCCViewer.MouseTool` objects, and are the known types returned `PCCViewer.MouseTool#getType`.

Type:

- string

Properties:

Name	Description
Magnifier : string	Use the magnifier mouse tool type to display a magnifying glass on left click and drag.
SelectToZoom : string	Use the select to zoom mouse tool type to select an area of the page to zoom in on.
Pan : string	Use the pan mouse tool type to drag the image up, down, left, or right.
PanAndEdit : string	Use the mouse or touch to drag the image up, down, left, or right. When clicking or touching over an annotation, the annotation will be selected, edited, moved, or resized, based on user actions.
SelectText : string	Use the select text mouse tool type to select text in the document.
EditMarks : string	Use the edit marks mouse tool type to select one or more marks (annotations and redactions) in the document. A mark can be clicked on for editing, or a rectangle can be drawn to select multiple marks.
LineAnnotation : string	Use the line annotation tool mouse tool type to draw a line annotation.
RectangleAnnotation : string	Use the rectangle annotation mouse tool type to draw a rectangle annotation.
EllipseAnnotation : string	Use the ellipse annotation mouse tool type to draw an ellipse annotation.
TextAnnotation : string	Use the text annotation mouse tool type to draw a text annotation.
StampAnnotation : string	Use the stamp annotation mouse tool type to draw a stamp annotation.
HighlightAnnotation : string	Use the highlight annotation mouse tool type to select text and create a highlight annotation.
TextHyperlinkAnnotation : string	Use the text hyperlink annotation mouse tool type to select text and create a text hyperlink annotation.
FreehandAnnotation : string	Use the freehand annotation tool mouse tool type to draw a freehand annotation.
RectangleRedaction : string	Use the rectangle redaction mouse tool type to draw a rectangle redaction.
TransparentRectangleRedaction : string	Use the transparent rectangle redaction mouse tool type to draw a transparent rectangle redaction.
TextRedaction : string	Use the text redaction mouse tool type to draw a text redaction.
TextInputSignature : string	Use the text input signature mouse tool type to draw a text input signature.
TextAreaSignature : string	Use the text area signature mouse tool type to draw a text area signature.
StampRedaction : string	Use the stamp redaction mouse tool type to draw a stamp redaction.

Name	Description
<code>PlaceSignature : string</code>	Use to click on the document and place a signature in that location.
<code>TextSelectionRedaction : string</code>	Use the TextSelectionRedaction mouse tool type to select text and create a text selection redaction.
<code>ImageStampAnnotation : string</code>	Use the ImageStampAnnotation mouse tool type to place ImageStamp annotation.
<code>ImageStampRedaction : string</code>	Use the ImageStampAnnotation mouse tool type to place ImageStamp redaction.
<code>PolylineAnnotation : string</code>	Use the PolylineAnnotation mouse tool type to place Polyline annotation.
<code>StrikethroughAnnotation : string</code>	Use the StrikethroughAnnotation mouse tool type to place Strikethrough annotation.

name :string

Gets the name of the mouse tool.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter method [PCCViewer.MouseTool#getName](#).

Type:

- string

See: [PCCViewer.MouseTool#getName](#)

Example

```
// get the mouse tool's name
var mouseToolName = myMouseTool.name;

// do something with the name
alert("Mouse tool name is " + mouseToolName);
```

templateMark :[PCCViewer.Mark](#)

Gets the template mark associated with an annotation or redaction mouse tool.

This property is defined on MouseTool objects that are annotation or redaction types: LineAnnotation, RectangleAnnotation, EllipseAnnotation, TextAnnotation, StampAnnotation, HighlightAnnotation, RectangleRedaction, TransparentRectangleRedaction, TextRedaction, TextInputSignature, StampRedaction, TextHyperlinkAnnotation, StrikethroughAnnotation, TextAreaSignature.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter

method [PCCViewer.MouseTool#getTemplateMark](#).

Type:

- [PCCViewer.Mark](#)

See: [PCCViewer.MouseTool#getTemplateMark](#)

Example

```
var myMouseTool =
PCCViewer.MouseTools.getMouseTool(mouseToolName);

// Check if templateMark is a property in the MouseTool
object
if (templateMark in myMouseTool) {
    // get the template mark
    var templateMark = myMouseTool.templateMark;

    // Do something with the template mark. For example,
    set the color.
    if (templateMark.setColor) {
        templateMark.setColor("#FF0000");
    }
}
```

type :string

Gets the type of the mouse tool.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter method [PCCViewer.MouseTool#getType](#).

Type:

- string

See: [PCCViewer.MouseTool.Type](#) for a list of possible mouse tool types.
[PCCViewer.MouseTool#getType](#)

Example

```
var myMouseTool =
PCCViewer.MouseTools.getMouseTool(mouseToolName);

// get the mouse tool's type
var mouseToolType = myMouseTool.type;
```

```
// do something with the type
switch (mouseToolType) {
    case PCCViewer.MouseTool.Type.LineAnnotation:
        ...
        break;
    default:
        ...
}
```

Methods

getName() → {string}

Gets the name of the mouse tool.

See: [PCCViewer.MouseTool#name](#)

Returns:

The name of the mouse tool.

Type
string

Example

```
// get the mouse tool's name
var mouseToolName = myMouseTool.getName();

// do something with the name
alert("Mouse tool name is " + mouseToolName);
```

getTemplateMark() → {PCCViewer.Mark}

Gets the template mark associated with an annotation or redaction mouse tool.

This method is defined on MouseTool objects that are annotation or redaction types: LineAnnotation, RectangleAnnotation, EllipseAnnotation, TextAnnotation, StampAnnotation, HighlightAnnotation, RectangleRedaction, TransparentRectangleRedaction, TextRedaction, TextInputSignature, StampRedaction, TextHyperlinkAnnotation, StrikethroughAnnotation, TextAreaSignature.

Returns:

The template mark for the mouse tool.

Type

[PCCViewer.Mark](#)

Example

```
var myMouseTool =
PCCViewer.MouseTools.getMouseTool(mouseToolName);

// Check if getTemplateMark is defined
if (myMouseTool.getTemplateMark) {
    // get the template mark
    var templateMark = myMouseTool.getTemplateMark();

    // Do something with the template mark. For example,
    set the color.
    if (templateMark.setColor) {
        templateMark.setColor("#FF0000");
    }
}
```

getType() → {string}

Gets the type of the mouse tool.

See: [PCCViewer.MouseTool.Type](#) for a list of possible mouse tool types.
[PCCViewer.MouseTool#type](#)

Returns:

The type of the mouse tool.

Type
string

Example

```
var myMouseTool =
PCCViewer.MouseTools.getMouseTool(mouseToolName);

// get the mouse tool's type
var mouseToolType = myMouseTool.getType();

// do something with the type
switch (mouseToolType) {
    case PCCViewer.MouseTool.Type.LineAnnotation:
        ...
        break;
    default:
```

```
    ...  
}
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: ObservableCollection

PCCViewer. ObservableCollection

`new ObservableCollection()`

Represents a dynamic collection that provides notifications when items get added or removed.

Members

(static, readonly) `EventType` :string

The known events for the collection.

Type:

- string

Properties:

Name	Description
<code>ItemAdded</code> : string	Triggered when an item is added to the collection.
<code>ItemRemoved</code> : string	Triggered when an item is removed from the collection.

Methods

`add(item)` → {[PCCViewer.ObservableCollection](#)}

Add an item to the collection.

Parameters:

Name	Type	Description
<code>item</code>	*	Any object or value to add to the collection.

Returns:

The collection object.

Type

[PCCViewer.ObservableCollection](#)

forEach(iterator, thisArg_{opt}) → **{PCCViewer.ObservableCollection}**

A method to iterate over all items in the collection. This method matches the spec for `Array.prototype.forEach`.

Parameters:

Name	Type	Attributes	Description
iterator	function PCCViewer.ObservableCollection~iterator		The function to execute for each item in the collection.
thisArg	*	<optional>	The Object to be used as <code>this</code> for the iterator function.

Throws:

If the `iterator` parameter is not a function.

Type

`TypeError`

Returns:

The collection object.

Type

[PCCViewer.ObservableCollection](#)

off(eventType, handler) → **{PCCViewer.ObservableCollection}**

Unsubscribe from an event triggered on the collection.

Parameters:

Name	Type	Description
eventType	string	The type of event being unsubscribed.
handler	function	The function that was used to subscribe to the event.

See:

[PCCViewer.ViewerControl#off](#)

Returns:

The collection object.

Type

[PCCViewer.ObservableCollection](#)

on(eventType, handler) → {[PCCViewer.ObservableCollection](#)}

Subscribe to an event triggered on the collection.

Parameters:

Name	Type	Description
eventType	string	The type of event being subscribed.
handler	function	The function to call when the event is triggered.

See:

[PCCViewer.ViewerControl#on](#)

[PCCViewer.ObservableCollection.EventType](#)

Returns:

The collection object.

Type

[PCCViewer.ObservableCollection](#)

remove(item) → {[PCCViewer.ObservableCollection](#)}

Remove an item from the collection.

Parameters:

Name	Type	Description
item	*	The item to be removed. This must be the same object that was added to the collection.

Returns:

The collection object.

Type

[PCCViewer.ObservableCollection](#)

removeAll() → {[PCCViewer.ObservableCollection](#)}

Removes all items from the collection.

Returns:

The collection object.

Type

[PCCViewer.ObservableCollection](#)

toArray() → {Array.<*>}

Generates an array of all of the items in the collection.

Returns:

An array of all items in the collection.

Type

Array.<*>

Type Definitions

iterator(item, index, array)

The iterator function for the [PCCViewer.ObservableCollection#forEach](#) method.

This function can also have an optional `this` argument, as defined in the [PCCViewer.ObservableCollection#forEach](#) method.

Parameters:

Name	Type	Description
item	*	The item from the collection.
index	Number	The index of the item from the collection.
array	Array	An array of all the items in the collection.

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: PrintRequest

PCCViewer. PrintRequest

(protected) new PrintRequest()

The `PrintRequest` object is created when printing the document. This constructor should not be used directly. Instead, a print request is created by `PCCViewer.ViewerControl#print`, and it is also made available through the `PCCViewer.EventType.PrintRequested` event.

Example

```
// A PrintRequest object is created by and returned from
the call to the print method
var printRequest = viewerControl.print();
```

Members**(static, readonly) EventType :string**

A list of events that can be triggered by the `PCCViewer.PrintRequest` object.

Type:

- string

Properties:

Name	Description
PrintPagePrepared : string	<p>Event triggered when a page has been prepared. This event is used to indicate print progress.</p> <p>Augmented properties of the <code>PCCViewer.Event</code> object for this event:</p> <ul style="list-style-type: none"> • <code>index {number}</code> Indicates the index of the page that was prepared in respect to <code>totalPages</code>. • <code>pageNumber {number}</code> Indicates the page number of the page that was prepared. This page number of the page in the document. • <code>totalPages {number}</code> Indicates the total number of pages that are being printed.
PrintCompleted : string	<p>Event triggered when print has completed, either due to a success, failure, or a cancel. This event does not indicate whether a user successfully printed the document, as they can still cancel the browser dialog, but rather that all pages were prepared successfully in the print request.</p> <p>Augmented properties of the <code>PCCViewer.Event</code> object for this event:</p> <ul style="list-style-type: none"> • none
PrintCancelled : string	<p>Event triggered if printing is cancelled during the preparation process.</p> <p>Augmented properties of the <code>PCCViewer.Event</code> object for this event:</p>

Name	Description
	<ul style="list-style-type: none"> • none
PrintFailed : string	<p>Event triggered if the printing process failed due to an error.</p> <p>Augmented properties of the PCCViewer.Event object for this event:</p> <ul style="list-style-type: none"> • none

See: [PCCViewer.PrintRequest#on](#)
[PCCViewer.PrintRequest#off](#)

(readonly) **options** :object

Gets a copy of the validated options object which is used by the print request.

The original options object may have been provided to the method [PCCViewer.ViewerControl#print](#). If no options object was provided to `print`, or the options object did not define all properties, then the returned object will represent the actual options used.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5.

Type:

- object

See: [PCCViewer.PrintRequest#getOptions](#)

(readonly) **pageCount** :number

This property gets the number of pages which were requested in the print request.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5.

Type:

- number

See: [PCCViewer.PrintRequest#getPageCount](#)

(readonly) **preparedCount** :number

This property gets the number of pages which have currently been prepared.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5.

Type:

- number

See: [PCCViewer.PrintRequest#getPreparedCount](#)

Methods

`cancel()`

Cancels the print request. This immediately stops the print progress, and no pages will be printed.

`getOptions() → {object}`

Gets a copy of the validated options object which is used by the print request.

The original options object may have been provided to the method [PCCViewer.ViewerControl#print](#). If no options object was provided to `print`, or the options object did not define all properties, then the returned object will represent the actual options used.

Returns:

A copy of the print options object which is used by this print request.

- `range` {string} A comma-separated string with all page numbers for the requested pages.
- `orientation` {string} The requested print orientation.
- `paperSize` {string} The requested size of the paper to print on.
- `margins` {string} Indicated whether `default` or `none` margins were used. See [PCCViewer.ViewerControl#print](#).
- `includeMarks` {boolean} Whether to print the document marks.
- `includeAnnotation` {boolean} Whether to print the document annotations.
- `includeRedactions` {boolean} Whether to print the document redactions.
- `includeComments` {string} Location to print comments.
- `includeReasons` {string} Location to print redaction reasons.
- `redactionViewMode` {string} Whether to print document content text underneath solid rectangle redactions and selection text redactions marks.

Type

object

`getPageCount() → {number}`

Gets the number of pages which were requested in the print request.

Returns:

The number of pages which were requested to print.

Type

number

`getPreparedCount() → {number}`

Gets the number of pages which have currently been prepared.

Returns:

The number of pages which have been prepared.

Type
number

off() → {PCCViewer.PrintRequest}

Remove event listeners from the `PrintRequest` object.

See: [PCCViewer.ViewerControl#off](#) for more on how it is used.
[PCCViewer.PrintRequest.EventType](#) for a list of events.

Returns:

The object on which this method was called.

Type
[PCCViewer.PrintRequest](#)

on() → {PCCViewer.PrintRequest}

Add event listeners to the `PrintRequest` object.

See: [PCCViewer.PrintRequest.EventType](#) for a list of events.
[PCCViewer.ViewerControl#on](#) for more detailed examples.

Returns:

The object on which this method was called.

Type
[PCCViewer.PrintRequest](#)

Example

```
var printRequest = viewerControl.print();
printRequest
  .on(PCCViewer.PrintRequest.EventType.PrintCompleted,
    function(ev) {
      alert("Print completed.");
    })
```

```
.on(PCCViewer.PrintRequest.EventType.PrintPagePrepared,  
    function(ev) {  
        alert("Print progress: " + 100 * (ev.index + 1) /  
ev.totalPages + "%");  
    });
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: Promise

PCCViewer. Promise

The [PCCViewer.Promise](#) object is an implementation of the Promises/A+ standard.

"A promise represents the eventual result of an asynchronous operation. The primary way of interacting with a promise is through its then method, which registers callbacks to receive either a promise's eventual value or the reason why the promise cannot be fulfilled." --

[Promises/A+ standard](#)

The PrizmDoc Viewing Client API uses Promises as a means for a caller to subscribe callbacks for an asynchronous operation. This API uses promises as an alternative to the pattern of providing callbacks as arguments to the API method.

The [PCCViewer.Promise](#) object is compatible with other Promises/A+ implementations, and other non-conformant promise implementations, which are "thenable" (i.e. the promise exposes a `.then()` method).

Constructor

`new Promise()`

The Promise constructor is for internal use only. Promise objects returned by other API methods are created with this constructor.

Methods

`(static) all(promises)`

Returns a promise that is fulfilled when all of the input promises are fulfilled. The returned promise is fulfilled with an array of the fulfilment values for all of the input promises.

If any of the promises are rejected, then the returned promise will be rejected with the reason of that rejected promise. If rejected, there will be guarantee of the state all promises in the promises array, some have been resolved and some may still be pending.

Parameters:

Name	Type	Description
promises	Array. <(PCCViewer.Promise thenable *)>	<p>An array of values that will be resolved. If a value is not a PCCViewer.Promise object, then this method will create a new PCCViewer.Promise and fulfill it with the value.</p> <p>Resolution of various types is as follows.</p> <ul style="list-style-type: none">• If the item is a PCCViewer.Promise, then the output value will be the fulfilment value of the promise.• If the item is thenable, then the output value will be the fulfilment value of the thenable.• Otherwise, the output value will be the item.

Throws:

If the `promises` argument is not an array.

Type
TypeError

Example

```
var viewerControl =
$("#myElement").pccViewer(...).viewerControl;

// Get page text for specific pages
PCCViewer.Promise.all([
  viewerControl.requestPageText(1),
  viewerControl.requestPageText(2)]).then(
  function onFulfilled(values) {
    // Values is an array that contains the text of
    pages 1 & 2.
    var page1Text = values[0];
    var page2Text = values[1];
  },
  function onRejected(error) {
    alert("Something went wrong getting the page text.
" + (error.message ? error.message : error));
  }
);

// Get attributes for all pages
var allPages = _.range(1, viewerControl.getPageCount() +
1); // Using Underscore.js - generates an array like [1, 2,
..., 12]
var pageAttributePromises = _.map(allPages,
viewerControl.requestPageAttributes, viewerControl); //
```

```

Using Underscore.js
PCCViewer.Promise.all(pageAttributePromises).then(
  function onFulfilled(allPageAttributes) {
    console.log(JSON.stringify(allPageAttributes));
  },
  function onRejected(error) {
    alert("Something went wrong getting the page
attributes. " + (error.message ? error.message : error));
  }
);

// it's OK to pass a value that is not a promise
PCCViewer.Promise.all([
  viewerControl.requestPageAttributes(1),
  true]).then(
  function(values) {
    // Values is an array that contains the text of
pages 1 and the value `true`.
    var page1Text = values[0]; // text of page 1
    var otherValue = values[1]; // true
  }
);

```

then(*onFulfilled_{opt}*, *onRejected_{opt}*) → {**PCCViewer.Promise**}

Use `.then(...)` to register callbacks to access the current or eventual value, or reason, of the promise.

A promise is in one of three states: pending, resolved, rejected. The `onFulfilled` callback will be called when a promise is resolved, or if a promise is already resolved, then the `onFulfilled` callback will be called immediately. The `onRejected` callback will be called when a promise is rejected, or if a promise is already rejected, then the `onRejected` callback will be called immediately.

Parameters:

Name	Type	Attributes	Description
<code>onFulfilled</code>	PCCViewer.Promise~onFulfilled	<optional>	Called if or when the promise is resolved. Optionally pass a value of <code>null</code> or <code>undefined</code> if you do not use this callback, but you want to provide an <code>onRejected</code> callback.
<code>onRejected</code>	PCCViewer.Promise~onRejected	<optional>	Called if or when the promise is rejected.

Returns:

A promise object that is resolved according to the Promises/A+ standard.

Type

[PCCViewer.Promise](#)

Example

```
var viewerControl =
$("#myElement").pccViewer(...).viewerControl;

// a basic example
viewerControl.requestPageText(1).then(
    function onFulfilled(value) {
        // according to the definition of requestPageText,
        the promise will be resolved with the text
        // of the page.
        var pageText = value;
    },
    function onRejected(error) {
        // according to the definition of requestPageText,
        the promise will be rejected if there is
        // an error extracting text for the document.
        alert("Something went wrong getting the text of
page 1. " + (error.message ? error.message : error));
    }
);

// it's OK to pass a value of null (or undefined) for
`onFulfilled`
viewerControl.requestPageText(1).then(
    null,
    function onRejected(error) { ... }
);

// it's OK to ignore the onRejected parameter, or pass null
or undefined
viewerControl.requestPageText(1).then(
    function onFulfilled(value) { ... }
);
```

Type Definitions

onFulfilled(value)

An `onFulfilled` callback is called if or when a `PCCViewer.Promise` is resolved.

Parameters:

Name	Type	Description
value	*	The type and value of the <code>value</code> argument depends on the API method that generated the Promise object. See the documentation for the method that generated the Promise.

`onRejected(reason)`

An `onRejected` callback is called if or when a `PCCViewer.Promise` is rejected.

Parameters:

Name	Type	Description
<code>reason</code>	*	The type and value of the <code>reason</code> argument depends on the API method that generated the Promise object. See the documentation for the method that generated the Promise.

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: Revision

`PCCViewer.Revision`

`new Revision()`

The `Revision` object is created when requesting the revisions for a document comparison. It represents a change identified in a document comparison and retrieved using [PCCViewer.ViewerControl#requestRevisions](#).

This constructor should not be used directly. Instead, only access revisions created by [PCCViewer.ViewerControl#requestRevisions](#), through the [PCCViewer.RevisionsRequest](#) object.

See:

- Use [PCCViewer.RevisionsRequest#getRevisions](#) to get results from a ``RevisionsRequest`` object.
- Use [PCCViewer.RevisionsRequest#revisions](#) to get results from a ``RevisionsRequest`` object.

Members

`(static, readonly) Type :string`

The `PCCViewer.Revision.Type` enumeration defines Revision Types known to the `ViewerControl`.

Type:

- `string`

Properties:

Name	Description
<code>ContentInserted : string</code>	Indicates that an insertion occurred.

Name	Description
<code>ContentDeleted</code> : string	Indicates that a deletion occurred.
<code>PropertyChanged</code> : string	Indicates that a property was changed.
<code>ParagraphNumberChanged</code> : string	Indicates that a paragraph number changed.
<code>FieldDisplayChanged</code> : string	Indicates that a field display changed.
<code>RevisionMarkedAsReconciledConflict</code> : string	Indicates that a revision was marked as reconciled conflict.
<code>RevisionMarkedAsConflict</code> : string	Indicates that a revision was marked as conflict.
<code>StyleChanged</code> : string	Indicates a style change.
<code>ContentReplaced</code> : string	Indicates that content was replaced.
<code>ParagraphPropertyChanged</code> : string	Indicates that a paragraph property changed.
<code>TablePropertyChanged</code> : string	Indicates that a table property changed.
<code>SectionPropertyChanged</code> : string	Indicates that a section property changed.
<code>StyleDefinitionChanged</code> : string	Indicates that a style definition changed.
<code>ContentMovedFrom</code> : string	Indicates that content was moved from this location.
<code>ContentMovedTo</code> : string	Indicates that content was moved to this location.
<code>TableCellInserted</code> : string	Indicates that a table cell was inserted.
<code>TableCellDeleted</code> : string	Indicates that a table cell was deleted.
<code>TableCellsMerged</code> : string	Indicates that table cells were merged.
<code>Unknown</code> : string	Indicates a revision of an unknown type.

(readonly) `id` :number

Gets the ID of the revision.

The ID is unique only to revisions in the current revisions request and may be repeated in later revisions requests.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- number

See: [PCCViewer.Revision#getId](#)

(readonly) `pageNumber` :number

Gets the page number of the document on which the revision ends.

If the revision is contained on a single page, this returns the page number of that page. If the revision spans multiple pages, this returns the page number of the last page.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- number

See: [PCCViewer.Revision#getEndPageNumber](#)

(readonly) **type** : [PCCViewer.Revision.Type](#)

Gets the type of the revision.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- [PCCViewer.Revision.Type](#)

See: [PCCViewer.Revision#getType](#)

Methods

[getEndPageNumber\(\)](#) → {number}

Gets the page number of the document on which the revision ends.

If the revision is contained on a single page, this returns the page number of that page. If the revision spans multiple pages, this returns the page number of the last page.

Returns:

The page number on which the revision ends.

Type
number

Example

```
var revisions = revisionsRequest.getRevisions();
var revision = revisions[0];
var pageNumber = revision.getEndPageNumber();
alert("Revision found on page: " + pageNumber);
```

[getId\(\)](#) → {number}

Gets the ID of the revision.

The ID is unique only to revisions in the current revisions request and may be repeated in later revisions requests.

See: [PCCViewer.RevisionsRequest#getRevisions](#)

Returns:

The ID of the revision.

Type
number

Example

```
var revisions = revisionsRequest.getRevisions();  
var revision = revisions[0];  
var id = revision.getId();
```

[getType\(\)](#) → {[PCCViewer.Revision.Type](#)}

Gets the type of the revision.

Returns:

The type of the revision.

Type
[PCCViewer.Revision.Type](#)

Example

```
var revisions = revisionsRequest.getRevisions();  
var revision = revisions[0];  
var revisionType = revision.getType();
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: RevisionsRequest

PCCViewer. RevisionsRequest

(protected) [new RevisionsRequest\(\)](#)

The `RevisionsRequest` object is created when requesting revisions for a document comparison. It triggers events to indicate revision retrieval progress and has properties to get the retrieved revisions and status.

This constructor should not be used directly. Instead, a revisions request is created by [PCCViewer.ViewerControl#requestRevisions](#).

Example

```
// A RevisionsRequest object is created by and returned
from the call to the requestRevisions method
var revisionsRequest = viewerControl.requestRevisions();
```

Members

(readonly) `errorCode` :number

Gets the error code if there was an error. If there was no error, `null` will be returned.

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- number

See: [PCCViewer.RevisionsRequest#getErrorCode](#)

(readonly) `errorMessage` :string

Returns a plain text, human-readable, fixed-local message that explains the error condition. If there was no error, `null` will be returned.

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- string

See: [PCCViewer.RevisionsRequest#getErrorMessage](#)

(readonly) `revisions` :Array.<[PCCViewer.Revision](#)>

Gets an array of all revisions produced by this `RevisionsRequest` up until this point.

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- [Array.<PCCViewer.Revision>](#)

See: [PCCViewer.RevisionsRequest#getRevisions](#)

Methods

[getErrorCode\(\)](#) → {number}

Returns the error code if there was an error. If there was no error, `null` will be returned.

Returns:

An error code indicating the type of error, or null.

Type
number

Example

```
var errorCode = revisionsRequest.getErrorCode();
```

[getErrorMessage\(\)](#) → {string}

Returns a plain text, human-readable, fixed-local message that explains the error condition. If there was no error, `null` will be returned.

Returns:

A plain text error message that explains the error condition, or null.

Type
string

Example

```
var errorMessage = revisionsRequest.getErrorMessage();
```

[getRevisions\(\)](#) → {Array.<PCCViewer.Revision>}

Returns an array of all revisions produced by this `RevisionsRequest` up until this point.

Returns:

An array of `Revision` objects. If no results are found, this will be an empty array.

Type

Array.<[PCCViewer.Revision](#)>

Example

```
var revisions = revisionsRequest.getRevisions();
```

off(eventType, handler) → {[PCCViewer.RevisionsRequest](#)}

Unsubscribe a handler from an event of the `RevisionsRequest`.

Typically, event is unsubscribed when you no longer want further notification of the event.

Parameters:

Name	Type	Description
eventType	string	A string specifying the event type. See PCCViewer.RevisionsRequest#on for possible values.
handler	function	The function that was previously subscribed to the event type.

Returns:

The `RevisionsRequest` object on which this method was called.

Type

[PCCViewer.RevisionsRequest](#)

Example

```
// subscribe
revisionsRequest.on(PCCViewer.EventType.RevisionsRetrievalCompleted,
onRevisionsRetrievalCompleted);

// unsubscribe
revisionsRequest.off(PCCViewer.EventType.RevisionsRetrievalCompleted,
onRevisionsRetrievalCompleted);

// handler declaration
function onRevisionsRetrievalCompleted(ev) {
    alert("Revisions retrieval completed! Number of
revisions: " + revisionsRequest.getRevisions().length);
}
```


`on(eventType, handler) → {PCCViewer.RevisionsRequest}`

Subscribe a handler to an event of the `RevisionsRequest`.

Parameters:

Name	Type	Description
<code>eventType</code>	<code>string</code>	A string that specifies the event type. <ul style="list-style-type: none">• "RevisionsRetrievalCompleted" - PCCViewer.EventType.RevisionsRetrievalCompleted• "RevisionsRetrievalFailed" - PCCViewer.EventType.RevisionsRetrievalFailed• "RevisionsAvailable" - PCCViewer.EventType.RevisionsAvailable• "PartialRevisionsAvailable" - PCCViewer.EventType.PartialRevisionsAvailable
<code>handler</code>	<code>function</code>	The function that will be called whenever the event is triggered.

Returns:

The `RevisionsRequest` object on which this method was called.

Type

[PCCViewer.RevisionsRequest](#)

Example

```
// subscribe
revisionsRequest.on(PCCViewer.EventType.RevisionsRetrievalCompleted,
onRevisionsRetrievalCompleted);

// handler declaration
function onRevisionsRetrievalCompleted(ev) {
    alert("Revisions retrieval completed! Number of
revisions: " + revisionsRequest.getRevisions().length);
}
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: SearchRequest

PCCViewer. SearchRequest

(protected) `new SearchRequest()`

The `SearchRequest` object is created when searching a document. It triggers events to indicate search progress and it has properties to get the search results and status.

This constructor should not be used directly. Instead, a search request is created by [PCCViewer.ViewerControl#search](#), and it is also made available through the [PCCViewer.EventType.SearchPerformed](#) event.

Example

```
// A SearchRequest object is created by and returned from
the call to the search method
var searchRequest = viewerControl.search("FooBar");
```

Members

(readonly) `errorCode` :number

Gets the error code if there was an error. If there was no error, `null` will be returned.

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- number

See: [PCCViewer.SearchRequest#getErrorCode](#)

(readonly) `errorMessage` :string

Returns a plain text, human-readable, fixed-local message that explains the error condition. If there was no error, `null` will be returned.

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- string

See: [PCCViewer.SearchRequest#getErrorMessage](#)

(readonly) `isComplete` :boolean

Gets a value (`true` or `false`) indicating if the search request is complete.

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- boolean

See: [PCCViewer.SearchRequest#getIsComplete](#)

(readonly) **results** :Array.<[PCCViewer.SearchResult](#)>

Gets an array of all search results produced by this `SearchRequest` up until this point.

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- Array.<[PCCViewer.SearchResult](#)>

See: [PCCViewer.SearchRequest#getResults](#)

(readonly) **searchQuery** :string

Gets the search query passed to [PCCViewer.ViewerControl#search](#).

If a string was passed to the `search` method, then this will return a [PCCViewer.ViewerControl~SearchQuery](#) object. The object will contain one search term (the provided string) and the options used for searching.

If an incomplete [PCCViewer.ViewerControl~SearchQuery](#) object was passed to `search`, then the object will be augmented with all options used for searching.

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- string

See: [PCCViewer.SearchRequest#getSearchQuery](#)

Methods

`cancel()`

Cancels the current search execution and triggers `SearchCancelled` event.

Example

```
searchRequest.cancel();
```

`getErrorCode()` → {number}

Returns the error code if there was an error. If there was no error, `null` will be returned.

Returns:

An error code indicating the type of error, or null.

The possible error codes are:

- 1010 - An unexpected exception occurred.
- 1011 - There was a failure retrieving data from the server.
- `ServerSearchUnavailable` - Server-side search is not available.

Type
number

Example

```
var errorCode = searchRequest.getErrorCode();
```

`getErrorMessage()` → {string}

Returns a plain text, human-readable, fixed-local message that explains the error condition. If there was no error, `null` will be returned.

Returns:

A plain text error message that explains the error condition, or null.

Type
string

Example

```
var errorMessage = searchRequest.getErrorMessage();
```

`getIsComplete()` → {boolean}

Returns a value (`true` or `false`) indicating if the search request is complete.

Returns:

A value indicating if search is complete.

Type
boolean

Example

```
var status = searchRequest.getIsComplete(); // true if
search is complete
```

getPagesWithoutText() → {Array.<number>}

Returns an array of page numbers that could not be searched because searchable text was not available for the page.

The set of pages without searchable text may still contain text embedded in a rasterized image, but the viewer is unable to detect or search this text. Therefore it is useful to notify the end user when some pages could not be searched.

Returns:

Returns an array of page numbers, or an empty array if all pages had searchable text.

Type

Array.<number>

Example

```
// Use pagesWithoutText to alert the end user that some
pages could not be search.
var pagesWithoutText = searchRequest.getPagesWithoutText();
```

getResults() → {Array.<PCCViewer.SearchResult>}

Returns an array of all search results produced by this `SearchRequest` up until this point.

Returns:

An array of `SearchResult` objects. If no results are found, this will be an empty array.

Type

Array.<PCCViewer.SearchResult>

Example

```
var searchResults = searchRequest.getResults();
```

getSearchQuery() → {PCCViewer.ViewerControl~SearchQuery}

Returns the search query passed to `PCCViewer.ViewerControl#search`.

If a string was passed to the `search` method, then this will return a [PCCViewer.ViewerControl~SearchQuery](#) object. The object will contain one search term (the provided string) and the options used for searching.

If an incomplete [PCCViewer.ViewerControl~SearchQuery](#) object was passed to `search`, then the object will be augmented with all options used for searching.

Returns:

The search query for this search request.

Type

[PCCViewer.ViewerControl~SearchQuery](#)

Example

```
var searchQuery = searchRequest.getSearchQuery();
```

off(eventType, handler) → {PCCViewer.SearchRequest}

Unsubscribe a handler from an event of the `SearchRequest`.

Typically, event is unsubscribed when you no longer want further notification of the event.

Parameters:

Name	Type	Description
eventType	string	A string specifying the event type. See PCCViewer.SearchRequest#on for possible values.
handler	function	The function that was previously subscribed to the event type.

Returns:

The `SearchRequest` object on which this method was called.

Type

[PCCViewer.SearchRequest](#)

Example

```
// subscribe
searchRequest.on(PCCViewer.EventType.SearchCompleted,
onSearchCompleted);

// unsubscribe
searchRequest.off(PCCViewer.EventType.SearchCompleted,
onSearchCompleted);
```

```
// handler declaration
function onSearchCompleted(ev) {
    alert("Search completed! Number of hits :" +
searchRequest.getResults().length);
}
```

on(eventType, handler) → {PCCViewer.SearchRequest}

Subscribe a handler to an event of the SearchRequest.

Parameters:

Name	Type	Description
eventType	string	A string that specifies the event type. <ul style="list-style-type: none">• "SearchCompleted" - PCCViewer.EventType.SearchCompleted• "SearchFailed" - PCCViewer.EventType.SearchFailed• "SearchCancelled" - PCCViewer.EventType.SearchCancelled• "SearchResultsAvailable" - PCCViewer.EventType.SearchResultsAvailable• "PartialSearchResultsAvailable" - PCCViewer.EventType.PartialSearchResultsAvailable
handler	function	The function that will be called whenever the event is triggered.

Returns:

The SearchRequest object on which this method was called.

Type

[PCCViewer.SearchRequest](#)

Example

```
// subscribe
searchRequest.on(PCCViewer.EventType.SearchCompleted,
onSearchCompleted);

// handler declaration
function onSearchCompleted(ev) {
    alert("Search completed! Number of hits :" +
searchRequest.getResults().length);
}
```

Class: SearchResult

PCCViewer. SearchResult

new SearchResult()

The `SearchResult` object is created when searching a document. It represents a "search hit", the text in the document that matched a search term in the `searchQuery`, which was passed to [PCCViewer.ViewerControl#search](#).

This constructor should not be used directly. Instead, only access search results created by [PCCViewer.ViewerControl#search](#), through the [PCCViewer.SearchRequest](#) object.

See: Use [PCCViewer.SearchRequest#getResults](#) to get results from a ``SearchRequest`` object.
Use [PCCViewer.SearchRequest#results](#) to get results from a ``SearchRequest`` object.

Members

(readonly) `boundingRectangle` :Object

Gets the bounding rectangle of the search result.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- Object

See: [PCCViewer.SearchResult#getBoundingRectangle](#)

(readonly) `context` :string

Gets the search result text and some surrounding text.

The number of character in the context before and after the search result text can be configured using the [PCCViewer.ViewerControl~SearchQuery](#) passed to the [PCCViewer.ViewerControl#search](#) method.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- string

See: [PCCViewer.SearchResult#getContext](#)

(readonly) **highlightColor** :string

Gets the highlight color of the search result, in hex notation (e.g. "#F1F1F1").

The highlight color can be specified in the [PCCViewer.ViewerControl~SearchQuery](#) passed to the [PCCViewer.ViewerControl#search](#) method. If a highlight color is not specified on the `searchQuery`, then a pseudo-random color is chosen.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- string

See: [PCCViewer.SearchResult#getHighlightColor](#)

(readonly) **id** :number

Gets the ID of the search result.

The ID is unique only to results in the current search request, and it may be repeated in later search requests.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- number

See: [PCCViewer.SearchResult#getId](#)

(readonly) **pageNumber** :number

Gets the page number of the document, on which the search result starts.

If the search result text is contained on a single page, this returns the page number of that page. If the search result text spans multiple pages, this returns the page number of the first page.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- number

See: [PCCViewer.SearchResult#getPageNumber](#)

(readonly) **searchTerm**

:[PCCViewer.ViewerControl~SearchTerm](#) | [PCCViewer.ViewerControl~ProximitySearchTerm](#)

Gets the search term object from the `searchQuery` object that was passed to the [PCCViewer.SearchTask](#) constructor.

If a string was passed to the `search` method, then this will return a [PCCViewer.ViewerControl~SearchTerm](#) object generated from the string.

The returned object will be augmented with all options used for searching.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- [PCCViewer.ViewerControl~SearchTerm](#) | [PCCViewer.ViewerControl~ProximitySearchTerm](#)

See: [PCCViewer.SearchResult#getSearchTerm](#)

(readonly) **searchTerm** :[PCCViewer.ViewerControl~SearchTerm](#)|[PCCViewer.ViewerControl~ProximitySearchTerm](#)

Gets the search term object from the `searchQuery` object that was passed to the [PCCViewer.ViewerControl#search](#) method.

If a string was passed to the `search` method, then this will return a [PCCViewer.ViewerControl~SearchTerm](#) object generated from the string.

The returned object will be augmented with all options used for searching.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- [PCCViewer.ViewerControl~SearchTerm](#) | [PCCViewer.ViewerControl~ProximitySearchTerm](#)

See: [PCCViewer.SearchResult#getSearchTerm](#)

(readonly) **startIndexInContext** :number

Gets the start index of the search result text within the context text returned by [PCCViewer.SearchTaskResult#getContext](#).

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- number

See: [PCCViewer.SearchTaskResult#getStartIndexInContext](#)

(readonly) **startIndexInContext** :number

Gets the start index of the search result text within the context text returned by [PCCViewer.SearchResult#getContext](#).

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

available in the older browsers like IE8.

Type:

- number

See: [PCCViewer.SearchResult#getStartIndexInContext](#)

(readonly) **text** :string

Gets the search result text. This is the text that matched the search query/term.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- string

See: [PCCViewer.SearchResult#getText](#)

Methods

getBoundingRectangle() → {Object}

Gets the bounding rectangle for the search result.

See: [PCCViewer.SearchResult#boundingRectangle](#)

Returns:

A rectangle object of the type {x: xValue, y: yValue, width: widthValue, height: heightValue}.

Type
Object

Example

```
if (searchResult.getBoundingRectangle) {  
    var boundingRectangle =  
    searchResult.getBoundingRectangle();  
}
```

getContext() → {string}

Gets the search result text and some surrounding text.

The number of character in the context before and after the search result text can be configured using the [PCCViewer.ViewerControl~SearchQuery](#) passed to the [PCCViewer.ViewerControl#search](#) method.

See: Use [PCCViewer.SearchResult#getStartIndexInContext](#) to identify the location of the search result text within the context.

Returns:

The context of the search result.

Type
string

Example

```
var results = searchRequest.getResults()
var result = results[0];

result.getText(); // e.g. "document"
result.getContext(); // e.g. "... the full
spectrum of document, content, & imaging s..."
result.getStartIndexInContext(); // e.g. 25
```

[getHighlightColor\(\)](#) → {string}

Gets the highlight color of the search result, in hex notation (e.g. "#F1F1F1").

The highlight color can be specified in the [PCCViewer.ViewerControl~SearchQuery](#) passed to the [PCCViewer.ViewerControl#search](#) method. If a highlight color is not specified on the `searchQuery`, then a pseudo-random color is chosen.

Returns:

The color of the search result highlight, in hexadecimal notation.

Type
string

Example

```
var results = searchRequest.getResults()
var result = results[0];
var highlightColor = result.getHighlightColor();
```

[getId\(\)](#) → {number}

Gets the ID of the search result.

The ID is unique only to results in the current search request, and it may be repeated in later search requests.

See: [PCCViewer.SearchRequest#getResults](#)

Returns:

The ID of the search result.

Type
number

Example

```
var results = searchRequest.getResults();
var result = results[0];
var id = result.getId();
```

[getPageNumber\(\)](#) → {number}

Gets the page number of the document, on which the search result starts.

If the search result text is contained on a single page, this returns the page number of that page. If the search result text spans multiple pages, this returns the page number of the first page.

Returns:

The page number on which the search result starts.

Type
number

Example

```
var results = searchRequest.getResults()
var result = results[0];
var pageNumber = result.getPageNumber();
alert("Search result found on page: " + pageNumber);
```

[getSearchTerm\(\)](#) →

{[PCCViewer.ViewerControl~SearchTerm](#) | [PCCViewer.ViewerControl~ProximitySearchTerm](#)}

Gets the search term object from the `searchQuery` object that was passed to the [PCCViewer.ViewerControl#search](#) method.

If a string was passed to the `search` method, then this will return a [PCCViewer.ViewerControl~SearchTerm](#) object generated from the string.

The returned object will be augmented with all options used for searching.

Returns:

The search term of the search result.

Type

[PCCViewer.ViewerControl~SearchTerm](#) | [PCCViewer.ViewerControl~ProximitySearchTerm](#)

Example

```
var results = searchRequest.getResults()
var result = results[0];
var searchTerm = result.getSearchTerm();
alert("This search result matched the search term: " +
searchTerm.searchTerm);
```

[getStartIndexInContext\(\)](#) → {number}

Gets the start index of the search result text within the context text returned by [PCCViewer.SearchResult#getContext](#).

Returns:

The start index of the search result text within the context text.

Type

number

Example

```
var results = searchRequest.getResults();
var result = results[0];
var startIndex = result.getStartIndexInContext();
```

[getStartIndexInPage\(\)](#) → {number}

Gets the start index of the search result text, within the entire text of the page that the result is on.

See:

Use [PCCViewer.ViewerControl#requestPageText](#) to get the full text of a page.
Use [PCCViewer.SearchResult#getText](#) to get the matched text of the search result.

Returns:

The start index of the matched text in the page.

Type
number

Example

```
var results = searchRequest.getResults()
var result = results[0];
var startIndex = result.getStartIndexInPage();
alert("The search result starts a character " + startIndex
+ " on the page.");
```

getText() → {string}

Gets the search result text. This is the text that matched the search query/term.

Returns:

The the search result text.

Type
string

Example

```
var results = searchRequest.getResults()
var result = results[0];
var text = result.getText();
alert("Search matched the text: " + text);
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: SearchTask

PCCViewer. SearchTask

This object represent a search task, which can be used to perform searches on any text string.

The PCCViewer.SearchTask.search method on the [PCCViewer.SearchTask](#) object can be used to search text contained in the Mark and comments objects. It will also perform search on any other text string.

Constructor

`new SearchTask(searchQuery)`

Creates a `SearchTask` object used for searching any text string.

Parameters:

Name	Type	Description
<code>searchQuery</code>	<code>string</code> PCCViewer.ViewerControl~SearchQuery	A value specifying the search query. The value specifies a single search term (string) or an object specifying multiple search terms and options. <i>NOTE: The searchQuery can be a single search term or a hash specifying one or more terms and options. If only a single search term (string) is supplied, then default options are used.</i>

See: [PCCViewer.SearchTask](#)
[PCCViewer.SearchTaskResult](#)

Throws:

- If search query is not a string or a valid [PCCViewer.ViewerControl~SearchQuery](#) object.

Type
Error

- When using the `SearchQuery` object, if the `searchQuery.searchTerm` is not an Array.

Type
Error

- When using the `SearchQuery` object, if the `searchQuery.searchTerms[i].searchTerm` property of each Object in the `searchTerms` array is not a string.

Type
Error

- If the combination of a search terms and matching options results in an invalid search, such as performing a wildcard search with only a `*` character and no valid content.

Type
Error

Example

```
// Search on multiple terms and specify options
```



```
var searchQuery = {
    searchTerms: [{
        searchTerm: "Full",
        contextPadding: 10,
        highlightColor: '#B22222',
        matchingOptions: {
            beginsWith: true
        }
    }]
};

// create a text annotation
var mark1 = viewerControl.addMark(1, "TextAnnotation");
// set text in the text annotation
mark1.setText("When Full-Text Search is being installed for
an existing client without Full-Text Search");
// create PCCViewer.SearchTask object
var searchTask = new PCCViewer.SearchTask(searchQuery);
// use the method PCCViewer.SearchTask.search to search the
word "Full" in the annotation text
var results = searchTask.search(mark1.getText());
// use it to search some other text string
var results2 = searchTask.search("To enable the full-text
search functionality, your system should have a dedicated
server.");
```

Methods

search(The) → {Array.<[PCCViewer.SearchTaskResult](#)>}

Searches any text string using the search criteria that were provided to the [PCCViewer.SearchTask](#) constructor.

Parameters:

Name	Type	Description
The	string	text string to be searched.

Returns:

An array of [PCCViewer.SearchTaskResult](#) objects.

Type

Array.<[PCCViewer.SearchTaskResult](#)>

Example

```
var searchQuery = {
  searchTerms: [{
    searchTerm: "client",
    contextPadding: 10,
    highlightColor: '#B22222',
    matchingOptions: {
      beginsWith: true,
    }
  }]
};
var textString = "When Full-Text Search is being installed
for an existing client without Full-Text Search";
var searchTask = new PCCViewer.SearchTask(searchQuery);
//search the textString
var results = searchTask.search(textString);
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: SearchTaskResult

PCCViewer. SearchTaskResult

new SearchTaskResult()

The `SearchTaskResult` object is created when searching a given text. It represents a "search hit", the text in the provided text that matched a search term in the `searchQuery`, which was passed to the method `PCCViewer.SearchTask.search`.

This constructor should not be used directly. Instead, only access search results created by the method `PCCViewer.SearchTask.search` method.

Members

(readonly) `context` :string

Gets the search result text and some surrounding text.

The number of character in the context before and after the search result text can be configured using the [PCCViewer.ViewerControl~SearchQuery](#) passed to the [PCCViewer.SearchTask](#) constructor.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- string

See: [PCCViewer.SearchTaskResult#getContext](#)

(readonly) **highlightColor** :string

Gets the highlight color of the search result, in hex notation (e.g. "#F1F1F1").

The highlight color can be specified in the [PCCViewer.ViewerControl~SearchQuery](#) passed to the [PCCViewer.SearchTask](#) constructor. If a highlight color is not specified on the `searchQuery`, then a pseudo-random color is chosen.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- string

See: [PCCViewer.SearchTaskResult#getHighlightColor](#)

(readonly) **id** :number

Gets the ID of the search result.

The ID is unique only to results in the current search request, and it may be repeated in later search requests.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- number

See: [PCCViewer.SearchTaskResult#getId](#)

(readonly) **text** :string

Gets the search result text. This is the text that matched the search query/term.

This is an ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Type:

- string

See: [PCCViewer.SearchTaskResult#getText](#)

Methods

getContext() → {string}

Gets the search result text and some surrounding text.

The number of character in the context before and after the search result text can be configured using the [PCCViewer.ViewerControl~SearchQuery](#) passed to the [PCCViewer.SearchTask#search](#) method.

See: Use [PCCViewer.SearchTaskResult#getStartIndexInContext](#) to identify the location of the search result text within the context.

Returns:

The context of the search result.

Type
string

Example

```
var mark1 = viewerControl.addMark(1, "TextAnnotation");
mark1.setText("When Full-Text Search is being installed for
an existing client without Full-Text Search");
var searchTask = new PCCViewer.SearchTask(searchQuery);
var results = searchTask.search(mark1.getText());
var result = results[0];

result.getText(); // e.g. "document"
result.getContext(); // e.g. "... the full
spectrum of document, content, & imaging s..."
result.getStartIndexInContext(); // e.g. 25
```

[getHighlightColor\(\)](#) → {string}

Gets the highlight color of the search result, in hex notation (e.g. "#F1F1F1").

The highlight color can be specified in the [PCCViewer.ViewerControl~SearchQuery](#) passed to the [PCCViewer.ViewerControl#search](#) method. If a highlight color is not specified on the `searchQuery`, then a pseudo-random color is chosen.

Returns:

The color of the search result highlight, in hexadecimal notation.

Type
string

Example

```
var mark1 = viewerControl.addMark(1, "TextAnnotation");
mark1.setText("When Full-Text Search is being installed for
an existing client without Full-Text Search");
var searchTask = new PCCViewer.SearchTask(searchQuery);
```

```
var results = searchTask.search(mark1.getText());
var result = results[0];
var highlightColor = result.getHighlightColor();
```

getId() → {number}

Gets the ID of the search result.

The ID is unique only to results in the current search request, and it may be repeated in later search requests.

Returns:

The ID of the search result.

Type

number

Example

```
var mark1 = viewerControl.addMark(1, "TextAnnotation");
mark1.setText("When Full-Text Search is being installed for
an existing client without Full-Text Search");
var searchTask = new PCCViewer.SearchTask(searchQuery);
var results = searchTask.search(mark1.getText());
var result = results[0];
var id = result.getId();
```

getSearchTerm() →

{[PCCViewer.ViewerControl~SearchTerm](#) | [PCCViewer.ViewerControl~ProximitySearchTerm](#)}

Gets the search term object from the `searchQuery` object that was passed to the [PCCViewer.SearchTask](#) constructor.

If a string was passed to the [PCCViewer.SearchTask](#) constructor, then this will return a [PCCViewer.ViewerControl~SearchTerm](#) object generated from the string.

The returned object will be augmented with all options used for searching.

Returns:

The search term of the search result.

Type

[PCCViewer.ViewerControl~SearchTerm](#) | [PCCViewer.ViewerControl~ProximitySearchTerm](#)

Example

```
var mark1 = viewerControl.addMark(1, "TextAnnotation");
mark1.setText("When Full-Text Search is being installed for
an existing client without Full-Text Search");
var searchTask = new PCCViewer.SearchTask(searchQuery);
var results = searchTask.search(mark1.getText());
var result = results[0];
var searchTerm = result.getSearchTerm();
alert("This search result matched the search term: " +
searchTerm.searchTerm);
```

getStartIndexInContext() → {number}

Gets the start index of the search result text within the context text returned by [PCCViewer.SearchResult#getContext](#).

Returns:

The start index of the search result text within the context text.

Type

number

Example

```
var mark1 = viewerControl.addMark(1, "TextAnnotation");
mark1.setText("When Full-Text Search is being installed for
an existing client without Full-Text Search");
var searchTask = new PCCViewer.SearchTask(searchQuery);
var results = searchTask.search(mark1.getText());
var result = results[0];
var startIndex = result.getStartIndexInContext();
```

getStartIndexInInput() → {number}

Gets the start index of the search result text, within the entire text string

See:

Use [PCCViewer.SearchTaskResult#getText](#) to get the matched text of the search result.

Returns:

The start index of the matched text in the provided text string to be searched.

Type

number

Example

```
var mark1 = viewerControl.addMark(1, "TextAnnotation");
mark1.setText("When Full-Text Search is being installed for
an existing client without Full-Text Search");
var searchTask = new PCCViewer.SearchTask(searchQuery);
var results = searchTask.search(mark1.getText());
var result = results[0];
var startIndex = result.getStartIndexInInput();
alert("The search result starts a character " + startIndex
+ " in the text string.");
```

getText() → {string}

Gets the search result text. This is the text that matched the search query/term.

Returns:

The search result text.

Type
string

Example

```
var mark1 = viewerControl.addMark(1, "TextAnnotation");
mark1.setText("When Full-Text Search is being installed for
an existing client without Full-Text Search");
var searchTask = new PCCViewer.SearchTask(searchQuery);
var results = searchTask.search(mark1.getText());
var result = results[0];
var text = result.getText();
alert("Search matched the text: " + text);
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Class: Signature Control

PCCViewer. SignatureControl

new SignatureControl(dom)

Creates a new signature drawing context in the given DOM element.

Parameters:

Name	Type	Description
dom	HTMLElement string	A DOM Element, or a string representing a valid query selector. This DOM element will be converted to a drawable area, so that the user can sign inside it. It can be styled any way you wish. However, this element must be visible on the page when the <code>SignatureControl</code> is initialized.

Throws:

- If the parameter passed in is not an HTML Element or a string.

Type
Error

- If the query selector string did not match any element.

Type
Error

Examples

```
// find the DOM element to use
var domElement = document.querySelector('#myDrawingArea');
// configure the control
var signatureControl =
PCCViewer.SignatureControl(domElement);
```

```
// shorthand to use the query selector directly
var signatureControl =
PCCViewer.SignatureControl('#myDrawingArea');
```

Methods

cancel() → `{PCCViewer.SignatureControl}`

Destroys the signature control and returns the `HTMLElement` back to its original state. Any drawn elements will be discarded and no `PCCViewer.Signatures~FreehandSignature` will be created.

Returns:

The `SignatureControl` that owns the method.

Type
`PCCViewer.SignatureControl`

clear() → {[PCCViewer.SignatureControl](#)}

Removes all drawn lines from the signature control.

Returns:

The `SignatureControl` that owns the method.

Type

[PCCViewer.SignatureControl](#)

done() → {[PCCViewer.Signatures~FreehandSignature](#)}

Creates a new [PCCViewer.Signatures~FreehandSignature](#) object from the elements drawn inside the `SignatureControl`. This method will also return the HTML Element that the `SignatureControl` was embedded in back to its original state.

If no content is drawn, the path returned by this method will be `M0, 0`, which is the shortest valid path which indicates that there is no content.

Returns:

The newly created signature.

Type

[PCCViewer.Signatures~FreehandSignature](#)

resize() → {[PCCViewer.SignatureControl](#)}

Reinitializes the size of the drawing area, so that the drawing area fits the entire size of the `HTML Element`. Content already drawn in the area will remain unchanged.

Returns:

The `SignatureControl` that owns the method.

Type

[PCCViewer.SignatureControl](#)

undo() → {[PCCViewer.SignatureControl](#)}

Removes the last drawn line from the signature control.

Returns:

The `SignatureControl` that owns the method.

Type

[PCCViewer.SignatureControl](#)

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:11 GMT-0400 (Eastern Daylight Time)

Class: SignatureDisplay

PCCViewer. SignatureDisplay

`new SignatureDisplay(domElement, signature) → {Object}`

Builds a DOM preview of the signature, to be used to display the signature outside of the viewer.

Parameters:

Name	Type	Description
<code>domElement</code>	<code>HTMLElement</code>	The element in which to insert signature preview.
<code>signature</code>	<code>Object</code> PCCViewer.Signatures~FreehandSignature PCCViewer.Signatures~TextSignature	The signature object to render in the preview.

Throws:

- If the `domElement` parameter is undefined or not a valid `HTMLElement`.
Type
`Error`
- If the `signature` parameter is undefined.
Type
`Error`
- If the `signature.path` property, in a `FreehandSignature` object, contains invalid data.
Type
`Error`
- If the `signature.text` property, in a `TextSignature` object, is not a string.

Type
Error

Returns:

An Object that contains the following properties:

- `width` {Number} The calculated width of the signature in pixels.
- `height` {Number} The calculated height of the signature in pixels.
- `clear` {Function} Remove all inserted content from the original HTML element.

Note: the width and height of a text signature will always be calculated using a 12 point font.

Type
Object

Example

```
// create a signature and a div
var signature = { path: "M0,0L100,0L100,100L0,100L0,0" };
var div = document.createElement('div');

// generate the signature preview
PCCViewer.SignatureDisplay(div, signature);

// display the div now
document.body.appendChild(div);
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:11 GMT-0400 (Eastern Daylight Time)

Class: ThumbnailControl

PCCViewer. ThumbnailControl

The `ThumbnailControl` is a viewer for the thumbnails associated with a document. It is associated with a [PCCViewer.ViewerControl](#) object at initialization, and will display the thumbnails of the document being viewed in that `ViewerControl`.

Constructor

`new ThumbnailControl(domElement, viewerControl)`

Creates a new `PCCViewer.ThumbnailControl` object.

Parameters:

Name	Type	Description
domElement	HTMLElement	The DOM element in which to embed the ThumbnailControl.
viewerControl	PCCViewer.ViewerControl	The ViewerControl object for which to display thumbnails.

See: [PCCViewer.ViewerControl](#)

Throws:

If either of the parameters is undefined or an invalid value.

Type
Error

Members

(static, readonly) **EventType** :string

The EventType enumeration defines event types known to [PCCViewer.ThumbnailControl](#).

Note: This enumeration is for convenience for API developers. Instead of using this enumeration, you can pass string values of the eventType (enumeration values)

Type:

- string

Properties:

Name	Description
PageSelectionChanged : string	Event is triggered when the collection of selected thumbnails changes. Augmented properties of the PCCViewer.Event object for this event: <ul style="list-style-type: none"> • <code>pageNumbers</code> {Array.<number>} The currently selected page numbers.

See: [PCCViewer.Event](#)
[PCCViewer.ThumbnailControl#on](#)
[PCCViewer.ThumbnailControl#off](#)

selectedPages :Array.<number>|number

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets and sets the selected pages in the Thumbnail Control.

Type:

- `Array.<number> | number`

See: [PCCViewer.ThumbnailControl#getSelectedPages](#)
[PCCViewer.ThumbnailControl#setSelectedPages](#)

(readonly) **viewerControl** : [PCCViewer.ViewerControl](#)

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets the [PCCViewer.ViewerControl](#) object associated with this `ThumbnailControl`.

Type:

- [PCCViewer.ViewerControl](#)

See: [PCCViewer.ThumbnailControl#getViewerControl](#)

(readonly) **visiblePages** : `Array.<number>`

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets the page numbers of the pages currently visible in the `ThumbnailControl` view.

Type:

- `Array.<number>`

See: [PCCViewer.ThumbnailControl#getVisiblePages](#)

Methods

`destroy()`

This method will remove the thumbnails from the original `HTMLElement` and return it to its previous state. It will destroy all of the thumbnails and release their resources. The `ThumbnailControl` can no longer be used after calling this method, and a new one will need to be created to continue viewing thumbnails.

`getSelectedPages()` → `{Array.<number>}`

Gets the currently selected thumbnails.

See: [PCCViewer.ThumbnailControl#setSelectedPages](#)

`PCCViewer.ThumbnailControl#selectedPages`

Returns:

The page numbers of the selected thumbnails.

Type

Array.<number>

`getViewerControl()` → `{PCCViewer.ViewerControl}`

Gets the `PCCViewer.ViewerControl` object associated with this `ThumbnailControl`.

Returns:

The `ViewerControl` associated with this `ThumbnailControl`.

Type

`PCCViewer.ViewerControl`

`getVisiblePages()` → `{Array.<number>}`

Gets the page numbers of the pages currently visible in the `ThumbnailControl` view. This will include pages that are only partly visible, as well as pages that are not yet fully loaded but part of their placeholder is visible.

Returns:

An array of page numbers.

Type

Array.<number>

`off(eventType, handler)` → `{PCCViewer.ThumbnailControl}`

Unsubscribe an event handler from a specified event type.

Typically, an event is unsubscribed when you no longer want further notification of the event.

Parameters:

Name	Type	Description
<code>eventType</code>	string	A string specifying the event type. See PCCViewer.ThumbnailControl.EventType for a list and description of all supported events.
<code>handler</code>	<code>PCCViewer.Event~eventHandler</code>	A function that was attached previously to the

Name	Type	Description
		ViewerControl. Note: This must be the same function object previously passed to PCCViewer.ThumbnailControl#on . It cannot be an different object that is functionally equivalent.

See: [PCCViewer.ViewerControl#on](#)
[PCCViewer.ViewerControl#off](#) for more details on usage.

Returns:

The `ThumbnailControl` object on which this method was called.

Type

[PCCViewer.ThumbnailControl](#)

on(eventType, handler) → {[PCCViewer.ThumbnailControl](#)}

Subscribe an event handler to an event of a specified type.

Parameters:

Name	Type	Description
eventType	string	A string that specifies the event type. This value is case-insensitive. See PCCViewer.ThumbnailControl.EventType for a list and description of all supported events.
handler	PCCViewer.Event~eventHandler	A function that will be called whenever the event is triggered.

See: [PCCViewer.ThumbnailControl#off](#)
[PCCViewer.ViewerControl#on](#) for more details on usage.

Returns:

The `ThumbnailControl` object on which this method was called.

Type

[PCCViewer.ThumbnailControl](#)

reflow() → {[PCCViewer.ThumbnailControl](#)}

This method will calculate the size of a thumbnail based on the defined CSS, and will fit each individual page element to the thumbnail container. This method can also be called to programmatically trigger the loading of newly visible pages in cases where the thumbnail view either dynamically or manually changes size, or the thumbnails themselves change size.

Note: This method can be a bit expensive, so it is best to debounce it from any event that triggers a thumbnail resize, such as a window resize.

Returns:

The `ThumbnailControl` object on which this method was called.

Type

`PCCViewer.ThumbnailControl`

Example

```
// First, get the thumbnailControl

// Create a resize function
var resize = function(){
    thumbnailControl.reflow();
};

// Create a debounce function
var debounceTimer;
var debounceEvent = function(){
    if (debounceTimer) {
        clearTimeout(debounceTimer);
        debounceTimer = undefined;
    }

    debounceTimer = setTimeout(resize, 300);
};

// Add the debounced function to the resize event
window.onresize = debounceEvent;
```

`scrollTo(pageNumber, optionsopt)` → `{PCCViewer.ThumbnailControl}`

Scrolls to a specified page. By default, this will be done using the minimum amount of scrolling, so pages that are above the current view will be scrolled to appear at the top of the view, and pages below the current view will be scrolled to appear at the bottom. This can be overridden with the `options` parameter.

Parameters:

Name	Type	Attributes	Description
<code>pageNumber</code>	<code>number</code>		The page to scroll to.
<code>options</code>	<code>Object</code>	<optional>	Provide options to the scroll method to indicate desired behavior.
Properties			
Name	Type	Attributes	Description

Name	Type	Attributes	Description				
			<table border="1"> <tr> <td>forceAlignTop</td> <td>boolean</td> <td><optional></td> <td>Overrides the default scrolling behavior, and forces the specified page to always be scrolled to the top of the view.</td> </tr> </table>	forceAlignTop	boolean	<optional>	Overrides the default scrolling behavior, and forces the specified page to always be scrolled to the top of the view.
forceAlignTop	boolean	<optional>	Overrides the default scrolling behavior, and forces the specified page to always be scrolled to the top of the view.				

Throws:

If the `pageNumber` is not within the range of the page count for the specified document.

Type
Error

Returns:

The `ThumbnailControl` object on which this method was called.

Type
[PCCViewer.ThumbnailControl](#)

`setSelectedPages(pageNumbers)` → **`{PCCViewer.ThumbnailControl}`**

Sets the currently selected thumbnails.

Parameters:

Name	Type	Description
<code>pageNumbers</code>	Array. <number> number	The page number (or numbers) to select. The first selected page will automatically be scrolled into view if it is not already visible. See PCCViewer.ThumbnailControl#scrollTo for more details on scrolling to a page.

See: [PCCViewer.ThumbnailControl#getSelectedPages](#)
[PCCViewer.ThumbnailControl#selectedPages](#)

Throws:

- If the `pageNumbers` parameter is undefined, or not a number or array of numbers.

Type
Error

- If any of the numbers defined in `pageNumbers` is out of range of the document page count.

Type
Error

Returns:

The `ThumbnailControl` object on which this method was called.

Type
[PCCViewer.ThumbnailControl](#)

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:11 GMT-0400 (Eastern Daylight Time)

Class: Viewer

PCCViewer. **Viewer**

This class gives programmatic access to the HTML 5 viewer.

Constructor

`new Viewer()`

Use the jQuery plugin [external:jQuery.fn#pccViewer](#) to create the viewer.

See: Use the jQuery plugin [external:jQuery.fn#pccViewer](#) to create an instance of this class.

Fires:

- [PCCViewer.Viewer#event:ViewerReady](#)

Example

```
// Create an instance of the class using the jQuery plugin
$("#mydiv").pccViewer(options); // returns PCCViewer.Viewer
instance
```

Members

`viewerControl` : [PCCViewer.ViewerControl](#)

Gets the [PCCViewer.ViewerControl](#) object used by the viewer instance. Through the

[PCCViewer.ViewerControl](#) object, the caller has API access to control the viewer behavior.

Type:

- [PCCViewer.ViewerControl](#)

Methods

destroy()

Cleans up the viewer DOM elements and leaves the elements as they were. This method also destroys the [PCCViewer.ViewerControl](#) object by calling [PCCViewer.ViewerControl#destroy](#) on the `viewerControl` that is associated with this viewer.

Example

```
var viewerPlugin = $('#mydiv').pccViewer(options);
var viewerControl = viewerPlugin.viewerControl;

viewerPlugin.destroy();
```

Events

ViewerReady

Fired when the viewer has initialized and is ready to be manipulated.

Type:

- [PCCViewer.Viewer](#)

Example

```
$('#mydiv').on('ViewerReady', function(pccViewer) {
    var marks = pccViewer.viewerControl.getAllMarks();
});

$('#mydiv').pccViewer(options);
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:11 GMT-0400 (Eastern Daylight Time)

Class: ViewerControl

PCCViewer. ViewerControl

The `ViewerControl` is a document viewer without any menus, buttons, dialogs, etc. The UI only consists of a page list, which allows a user to scroll through the pages of a document. It is the most basic viewer which shows a document. It can be used alone, or it can be augmented with UI elements (chrome) to expose more functionality to the end user.

Our out-of-the-box HTML5 viewer uses the `ViewerControl` to build a desktop and mobile ready viewer, with extensive UI chrome and a responsive design.

The `ViewerControl` has an API, which covers the full set of viewer functionality. Code that directly uses the `ViewerControl` will typically create UI elements - buttons, menus, and inputs - for the end user to interact with. These UI elements will be hooked up to call the `ViewerControl` API when the user interacts with the UI elements (e.g. on button click, call `.changeToNextPage()`).

When the `ViewerControl` is instantiated it fires initialization events in the following order:

- [PCCViewer.EventType.ViewerReady](#)
- [PCCViewer.EventType.PageCountReady](#)

NOTE: do not use the `ViewerControl` API until [PCCViewer.EventType.ViewerReady](#) has fired.

When viewing session is changed with [PCCViewer.ViewerControl#changeViewingSession](#) method call, `ViewerControl` fires events in the following order:

- [PCCViewer.EventType.ViewingSessionChanging](#)
- [PCCViewer.EventType.ViewingSessionChanged](#)
- [PCCViewer.EventType.PageCountReady](#)

NOTE: once the [PCCViewer.EventType.ViewingSessionChanging](#) event has fired, do not use the `ViewerControl` API until [PCCViewer.EventType.ViewingSessionChanged](#) event fires.

The `ViewerControl` also permits mouse and touch interaction. The behavior of the mouse tool or touch is set using the API method [PCCViewer.ViewerControl#setCurrentMouseTool](#). Once the tool is set, the user can interact with the viewer using the mouse tool or touch.

Constructor

`new ViewerControl(element, options)`

Creates a new `PCCViewer.ViewerControl` object.

Parameters:

Name	Type	Description
<code>element</code>	<code>HTMLDivElement</code>	Embed the <code>ViewerControl</code> in this element.
<code>options</code>	PCCViewer.ViewerControl~ViewerControlOptions	Specify options for the <code>ViewerControl</code> object. Some options are required.

Members

(readonly) **atMaxScale** :boolean

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

It checks whether the viewer is currently at the maximum zoom level.

Type:

- boolean

See: [PCCViewer.ViewerControl#getAtMaxScale](#)

(readonly) **atMinScale** :boolean

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

It checks whether the viewer is currently at the minimum zoom level.

Type:

- boolean

See: [PCCViewer.ViewerControl#getAtMinScale](#)

currentMouseTool :string

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets and sets the current mouse tool of the viewer by name.

Type:

- string

See: [PCCViewer.ViewerControl#getCurrentMouseTool](#)

[PCCViewer.ViewerControl#setCurrentMouseTool](#)

(readonly) **isCommentsPanelOpen** :boolean

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets a value indicating whether the comments panel is open.

Type:

- boolean

See: [PCCViewer.ViewerControl#getIsCommentsPanelOpen](#)

markHandleMode : [PCCViewer.RedactionViewMode](#)

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets or sets the mark handle mode.

Type:

- [PCCViewer.RedactionViewMode](#)

See: [PCCViewer.ViewerControl#getMarkHandleMode](#)
[PCCViewer.ViewerControl#setMarkHandleMode](#)
[PCCViewer.MarkHandleMode](#)

(readonly) **pageCount** : number

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets the document page count.

Type:

- number

See: [PCCViewer.ViewerControl#getPageCount](#)

pageNumber : number

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets and sets the current page of the viewer to the specified page number in the document. Setting the page number to a value other than the current page number will cause the viewer to navigate to the page number provided in the parameter.

Type:

- number

See: [PCCViewer.ViewerControl#getPageNumber](#)
[PCCViewer.ViewerControl#setPageNumber](#)

redactionViewMode : [PCCViewer.RedactionViewMode](#)

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets or sets the redaction view mode. This defines whether the document content text underneath

redaction rectangles needs to be visible in a "Draft" mode [PCCViewer.RedactionViewMode](#) enumerable values.

Type:

- [PCCViewer.RedactionViewMode](#)

See: [PCCViewer.ViewerControl#getRedactionViewMode](#)
[PCCViewer.ViewerControl#setRedactionViewMode](#)
[PCCViewer.RedactionViewMode](#)

scaleFactor :number

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets or sets the scale factor to use in the viewer, with 1 being 100% zoom.

Type:

- number

See: [PCCViewer.ViewerControl#getScaleFactor](#)
[PCCViewer.ViewerControl#setScaleFactor](#)

(readonly) **searchRequest** :[PCCViewer.SearchRequest](#)

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets the `SearchRequest` object from the last call to [PCCViewer.ViewerControl#search](#).

Type:

- [PCCViewer.SearchRequest](#)

See: [PCCViewer.ViewerControl#getSearchRequest](#)

selectedConversation :[PCCViewer.Conversation](#)

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets and sets the selected conversation.

Type:

- [PCCViewer.Conversation](#)

See: [PCCViewer.ViewerControl#getSelectedConversation](#)

[PCCViewer.ViewerControl#setSelectedConversation](#)

(readonly) **selectedMarks** :Array.<[PCCViewer.Mark](#)>

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets an array of selected marks.

Type:

- Array.<[PCCViewer.Mark](#)>

See: [PCCViewer.ViewerControl#getSelectedMarks](#)

selectedSearchResult :[PCCViewer.SearchResult](#)|null

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets and sets the selected `SearchResult` object. Returns null if no search results are selected. **Note:** Setting the search result through this property will always scroll to it.

Type:

- [PCCViewer.SearchResult](#) | null

See: [PCCViewer.ViewerControl#getSelectedSearchResult](#)
[PCCViewer.ViewerControl#setSelectedSearchResult](#)

viewMode :[PCCViewer.ViewMode](#)

An ECMA 5 accessor property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8.

Gets or sets the view mode. This defines how the document pages will be scaled, based on the values of the [PCCViewer.ViewMode](#) enumerable values.

Type:

- [PCCViewer.ViewMode](#)

See: [PCCViewer.ViewerControl#getViewMode](#)
[PCCViewer.ViewerControl#setViewMode](#)
[PCCViewer.ViewMode](#)

(inner) **TextSelection**

A plain object convention describing a text selection.

Properties:

Name	Description
<code>pageNumber</code> : number	The page number that the selection starts on.
<code>length</code> : number	The length of the text.
<code>text</code> : string	The selected text. This is plain text without any formatting.
<code>startIndex</code> : number	The index at which the selection starts in the page.
<code>rectangles</code> : Array. < PCCViewer.ViewerControl~TextSelectionRectangle >	An array of all the rectangles that make up the text selection.

(inner) TextSelectionRectangle

A plain object convention describing a text selection rectangle.

Properties:

Name	Description
<code>x</code> : number	The x-coordinate of the top-left corner of the rectangle.
<code>y</code> : number	The y-coordinate of the top-left corner of the rectangle.
<code>width</code> : number	The width of the rectangle.
<code>height</code> : number	The height of the rectangle.
<code>pageNumber</code> : number	The page number of the rectangle.

Methods

`addMark(pageNumber, markType) → {PCCViewer.Mark}`

Creates a new mark of a specific type and adds to the specified page.

Parameters:

Name	Type	Description
<code>pageNumber</code>	number	Indicates the page to which to add the mark.
<code>markType</code>	PCCViewer.Mark.Type string	Indicates the type of mark being added

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type
Error

- If an invalid `pageNumber` is provided.

Type
Error

Returns:

The new mark.

Type
[PCCViewer.Mark](#)

Example

```
viewerControl.addMark(1, "LineAnnotation");
```

`addMarkFromSearchResult(searchResult, markType)` → `{PCCViewer.Mark}`

Creates a new mark of a specific type and adds to the location where the specified search result is.

Parameters:

Name	Type	Description
<code>searchResult</code>	PCCViewer.SearchResult	Indicates the search result that will include the location for the new mark.
<code>markType</code>	PCCViewer.Mark.Type string	Indicates the type of mark being added.

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type
Error

- If an invalid `searchResult` is provided.

Type
Error

- If an invalid text based mark type is provided.

Type
Error

Returns:

The new mark.

Type
[PCCViewer.Mark](#)

Example

```
var requestObject = PCCViewer.search('Con');
var marks = [];
var newMark;
requestObject.on(PCCViewer.EventType.SearchCompleted,
function (event) {
    var searchResults = event.completedSearchResults;
    for (var i = 0; i < searchResults.length; i++) {
        newMark =
viewer.addMarkFromSearchResult(searchResults[i],
PCCViewer.Mark.Type.TextSelectionRedaction);
        marks.push(newMark);
    }
});
```

burnMarkup(options_{opt}) → {[PCCViewer.BurnRequest](#)}

Burns redactions and signatures in the document. **Note:** CAD files are not supported.

If the parameter `options.removeFormFields` is invalid, then the `PCCViewer.Promise` object that is returned will be rejected with the reason set to a `PCCViewer.Error` object with its code property set to `InvalidArgument`.

Parameters:

Name	Type	Attributes	Description		
options	Object	<optional>	This optional parameter specifies burn options to be used for burning the document.		
Properties					
Name	Type	Attributes	Default	Description	
marks	PCCViewer.Mark Array. <PCCViewer.Mark>	<optional>		Indicates the presence of marks in the document.	
burnSignatures	boolean	<optional>	true	Indicates whether burn signatures are present in the document.	
burnRedactions	boolean	<optional>	true	Indicates whether burn redactions are present in the document.	
burnAnnotations	boolean	<optional>	false	Indicates whether burn annotations are present in the document.	
filename	string	<optional>		Sets the filename of the document. If not specified, the default filename will be used.	
removeFormFields	Array.<string>	<optional>		Specifies the names of the form fields to be removed from the document. Only supported if the document is not redacted.	
redactionOptions	Object	<optional>		The properties of the redaction options. <ul style="list-style-type: none"> 	

Name	Type	Attributes	Description

See: [PCCViewer.BurnRequest](#) for more details on interacting with the burn process.

Returns:

A result `BurnRequest` for this task.

Type

[PCCViewer.BurnRequest](#)

Example

```
function onSuccessfullBurn (burnturl) {  
    alert("burntURL = " + burnturl);  
    console.log (burnturl);  
}
```

```
function onFailedBurn(error) {
    alert("burn Process failed, error:" + (error.message ?
error.message : error));
}

// A BurnRequest object is created by and returned from
the call to the burnMarkup method
var burnRequest = viewerControl.burnMarkup();
burnRequest.then(onSuccessfulBurn, onFailedBurn);

//register some events
burnRequest
    .on(PCCViewer.BurnRequest.EventType.BurnCompleted,
        function(ev) {
            alert("Document burn completed.");
        })
    .on(PCCViewer.BurnRequest.EventType.BurnProgress,
        function(event) {
            alert("Burn progress: " + event.percent + "%");
        })
    .on(PCCViewer.BurnRequest.EventType.BurnFailed,
        function(event) {
            alert("Document burn failed.");
        });

// Also, methods on the burnRequest object can be used

// get the options used to burn the document.
var optionsUsed = burnRequest.getOptions();

//if the process is still incomplete, cancel can be
used to stop queries to the server.
if (burnRequest.getProgress() >= 0 &&
burnRequest.getProgress() < 100) {
    burnRequest.cancel();
}
```

`canPrintMarks()` → `{boolean}`

Informs whether the current browser is capable of printing the annotations and redactions on a document. If `true`, the document can be printed with annotations and redactions. If `false`, the document will be printed without annotations, regardless of the print request made.

Returns:

A value indicating whether the browser is capable of printing annotations and redactions on a document.

Type

boolean

Example

```
var printWithMarks = true;
viewerControl.print({
  includeMarks: (printWithMarks &&
viewerControl.canPrintMarks())
});
```

`changeToFirstPage()` → `{PCCViewer.ViewerControl}`

Sets the current page of the viewer to the first page of the document.

Note: Does nothing if the current page is the first page of the document.

Returns:

The `ViewerControl` object on which this method was called.

Type

`PCCViewer.ViewerControl`

Example

```
viewerControl.changeToFirstPage();
```

`changeToLastPage()` → `{PCCViewer.ViewerControl}`

Sets the current page of the viewer to the last known page of the document.

Note: Does nothing if the current page is the last known page of the document.

Returns:

The `ViewerControl` object on which this method was called.

Type

`PCCViewer.ViewerControl`

Example

```
viewerControl.changeToLastPage();
```

`changeToNextPage()` → `{PCCViewer.ViewerControl}`

Sets the current page of the viewer to the next page of the document.

Note: Does nothing if the current page is the last known page of the document.

Returns:

The `ViewerControl` object on which this method was called.

Type

`PCCViewer.ViewerControl`

Example

```
viewerControl.changeToNextPage();
```

`changeToPrevPage()` → `{PCCViewer.ViewerControl}`

Sets the current page of the viewer to the previous page of the document.

Returns:

The `ViewerControl` object on which this method was called.

Type

`PCCViewer.ViewerControl`

Example

```
viewerControl.changeToPrevPage();
```

`changeViewingSession(viewingSessionId, isRestorable)` → `{PCCViewer.ViewerControl}`

Changes the viewer to a different viewing session, allowing the viewer to work with a different document. If you intend to return to the current viewing session in the future, set `isRestorable` to `true`. Otherwise, the current viewer state and any annotation data in memory will be disposed.

Parameters:

Name	Type	Description
<code>viewingSessionId</code>	<code>string</code>	<code>viewingSessionId</code> for the new viewing session to switch to, changing the document in use for the end user.
<code>isRestorable</code>	<code>boolean</code>	When <code>true</code> , unsaved changes to markup will be kept in memory and automatically restored if you change back to the current viewing session. When <code>false</code> , unsaved changes to markup will be discarded.

Throws:

- If the `PCCViewer.EventType.ViewerReady` event has not fired prior to using this method.

Type
Error

- If the web application has already called `changeViewingSession` (or `ViewingSessionChanging` has fired in response to user activity in the viewer) but corresponding `ViewingSessionChanged` event has not fired prior to using this method.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
`PCCViewer.ViewerControl`

Example

```
viewerControl.changeViewingSession(anotherViewingSessionId)

    .on(PCCViewer.EventType.ViewingSessionChanged,
function(ev) {
    alert("Viewing session changed.");
});
```

`clearMouseSelectedText(textSelection)` → `{PCCViewer.ViewerControl}`

Deselects the text provided in a `TextSelected` event.

Parameters:

Name	Type	Description
<code>textSelection</code>	<code>PCCViewer.ViewerControl~TextSelection</code>	The <code>textSelection</code> object provided in the <code>TextSelected</code> event arguments.

Throws:

- If `textSelection` is undefined.

Type
Error

- If `textSelection.pageNumber` is not a known page number.

Type
Error

- If `textSelection.startIndex` is not a number or is negative.

Type
Error

- If `textSelection.length` is not a number or is less than 1.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
var onTextSelected = function(ev) {
    var mark =
viewerControl.addMark(ev.textSelection.pageNumber,
'HighlightAnnotation');
    mark.setPosition(ev.textSelection);

    // clear the highlighted text

ViewerControl.clearMouseSelectedText(ev.textSelection);
};

viewerControl.on('TextSelected', onTextSelected);
```

`clearSearch()` → [{PCCViewer.ViewerControl}](#)

Clears the search hit highlights and removes the `SearchRequest` from the `ViewerControl`.

After calling this, [PCCViewer.ViewerControl#getSearchRequest](#) will not return the last `SearchRequest`.

Throws:

If the [PCCViewer.EventType.ViewerReady](#) event has not fired prior to using this method.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
viewerControl.search("Foo");  
// As search results become available, they are  
highlighted on the document.  
  
// Clear search result highlights from the document.  
viewerControl.clearSearch();
```

[clearSelectedSearchResult\(\)](#) → [{PCCViewer.ViewerControl}](#)

This methods clears the search result selection, or in other words, it deselects the search result. If there is not a selected search result when this method is called, then the method has no effect.

This method is offered as a convenience to API callers, who could also call [PCCViewer.ViewerControl#setSelectedSearchResult\(null\)](#).

See: [PCCViewer.ViewerControl#setSelectedSearchResult](#)
[PCCViewer.ViewerControl#clearSearch](#)

Throws:

If the [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
var searchRequest = viewerControl.search('Accusoft');

// add events to the search request
searchRequest.on("SearchCompleted", function(){
    // get the search results
    results = searchRequest.getResults();

    // set the result to the first
    viewerControl.setSelectedSearchResult(results[0],
true);

    // clear the selected search result
    viewerControl.clearSelectedSearchResult();
});
```

clientSearch(searchQuery) → {PCCViewer.SearchRequest}

Searches the text of the document for the given `searchQuery`. The search is performed client-side, which requires requesting from the server text for each page. This is efficient for smaller documents, but for large documents it is more efficient to use the [PCCViewer.ViewerControl#serverSearch](#) method instead.

This query can be a single search term or a hash specifying one or more terms and options. If only a single search term (string) is supplied, then default options are used.

Search completes asynchronously. The returned [PCCViewer.SearchRequest](#) object, provides events for search progress and members to access search results.

Parameters:

Name	Type	Description
searchQuery	string PCCViewer.ViewerControl~SearchQuery	A value specifying the search query. The value specifies a single search term (string) or an object specifying multiple search terms and options.

See: [PCCViewer.SearchRequest](#)
[PCCViewer.SearchResult](#)

Returns:

An object that represents the search request. This object allows the calling code to subscribe to search progress events and access search results.

Type
[PCCViewer.SearchRequest](#)

Examples

```
// Search on a single term with default options
var searchRequest = viewerControl.clientSearch("Hello");

// Subscribe to the PartialSearchResultsAvailable event
to get search results as they become available.
searchRequest.on('PartialSearchResultsAvailable',
function(_event) {
    // Get the newly available search results.
    var newResults = _event.partialSearchResults;
});

// Search on multiple terms and specify options
var searchQuery = {
    searchTerms: [{
        searchTerm: "sub",
        contextPadding: 25,
        highlightColor: '#B22222',
        matchingOptions: {
            beginsWith: true,
        }
    },
    {
        searchTerm: "Accusoft"
    }
    ]
};

var requestObject =
viewerControl.clientSearch(searchQuery);

//subscribe to the search request
requestObject.on('PartialSearchResultsAvailable',
function(_event) {
    var newResults = [];
    //Retrieve results
    newResults = _event.partialSearchResults;
});
```

closeCommentsPanel() → **{PCCViewer.ViewerControl}**

Closes the comments panel.

Throws:

If **PCCViewer.EventType.ViewerReady** event was not fired prior to using this method.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)

Example

```
viewerControl.closeCommentsPanel();
```

convertDocument()

Deprecated since v11.0 (use the [PCCViewer.ViewerControl#requestDocumentConversion](#) method instead).

convertPageToWindowCoordinates(pageNumber, points) → {Array.<Object>|Object}

Converts page-based coordinates to the current window coordinates. This allows passing in a coordinate point or an array of points as used in the `ViewerControl` API, in order to get the position of that point or points relative to the window.

Parameters:

Name	Type	Description									
pageNumber	number	A known page number to the viewer.									
points	Array.<Object> Object	A point or array of points in page coordinates. Each point will have the following properties: Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>x</td><td>number</td><td>the x page coordinate</td></tr><tr><td>y</td><td>number</td><td>the y page coordinate</td></tr></tbody></table>	Name	Type	Description	x	number	the x page coordinate	y	number	the y page coordinate
Name	Type	Description									
x	number	the x page coordinate									
y	number	the y page coordinate									

Throws:

- If the [PCCViewer.EventType.ViewerReady](#) event has not fired prior to calling this method.

Type

Error

- If `pageNumber` is not a currently known page.

Type

Error

- If any of the points do not have a valid `x` and `y` numeric parameters.

Type

Error

Returns:

A window point or an array of window points, depending on how the method was called. These points will be in the same order as the points passed into the function. Each window point will have the following parameters:

- `clientX` {number} The x window coordinate in pixels.
- `clientY` {number} The y window coordinate in pixels.

Type

Array.<Object> | Object

`convertToHighlight()`

Deprecated since v13.16 (use the [PCCViewer.ViewerControl#addMarkFromSearchResult](#) method instead).

`convertToRedaction()`

Deprecated since v13.16 (use the [PCCViewer.ViewerControl#addMarkFromSearchResult](#) method instead).

`copyMarks(marks) → {Array.<PCCViewer.Mark>}`

Makes a copy of the specified marks.

Note: This method requires that attributes of each page referenced by the marks have been obtained by the viewer prior to calling. Use [PCCViewer.ViewerControl#requestPageAttributes](#) to obtain the necessary page attributes before calling this method.

Parameters:

Name	Type	Description
marks	Array.<PCCViewer.Mark>	An array of marks to copy.

Throws:

If marks is not an array of marks known to the viewer.

Type

Error

Returns:

An array of the copied marks.

Type

Array.<PCCViewer.Mark>

Example

```
// Make a copy of the selected marks.  
markupLayer1.copyMarks (viewerControl.getSelectedMarks ());
```

deleteAllMarks() → {PCCViewer.ViewerControl}

Deletes all marks in all the pages of the document.

Returns:

The ViewerControl object on which this method was called.

Type

PCCViewer.ViewerControl

Example

```
viewerControl.deleteAllMarks ();
```

deleteMarks(marks) → {PCCViewer.ViewerControl}

Deletes the specified marks.

Parameters:

Name	Type	Description
marks	Array.<PCCViewer.Mark>	An Array of objects of type PCCViewer.Mark.

Throws:

- If PCCViewer.EventType.ViewerReady event was not fired prior to using this method.

Type

Error

- If any of the marks passed in are not valid objects of the type [PCCViewer.Mark](#).

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
// delete all selected marks  
viewerControl.deleteMarks (viewer.getSelectedMarks ());
```

deselectAllMarks() → [{PCCViewer.ViewerControl}](#)

Deselects all previously selected marks.

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
viewerControl.deselectAllMarks ();
```

deselectMarks(marks) → [{PCCViewer.ViewerControl}](#)

Deselects the marks provided in the parameter array object.

Parameters:

Name	Type	Description
marks	Array. < PCCViewer.Mark >	An array of PCCViewer.Mark objects that exist in the document and need to be deselected.

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to calling this method.

Type
Error

- If any of the mark objects are not valid [PCCViewer.Mark](#) objects, the id of the mark provided does not match the id of mark loaded in the viewer, or the mark provided was not previously added to the document pages.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
// deselect all marks with an odd-number ID
viewerControl.getSelectedMarks().forEach(function(mark) {
  var arr = [];
  if (+mark.getId() % 2) arr.push(mark);
  viewerControl.deselectMarks(arr);
});
```

deserializeMarks(values)

Deserializes JSON or a JSON-like object to [PCCViewer.Mark](#) objects, and adds them to the `ViewerControl`. This will display all deserialized marks.

Parameters:

Name	Type	Description
values	String.<JSON> Object Array.<Object>	The value to deserialize.

See: [PCCViewer.ViewerControl#serializeMarks](#)

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type
Error

- If the `json` parameter as a string is not a valid JSON string.

Type
Error

- If any mark has an unknown page number in the `pageNumber` property. *Note that if marks are being added before the `PageCountReady` event fires, this method will only know about page 1, and throw an error if adding marks to other pages.*

Type
Error

- If any mark has an unknown type in the `type` property.

Type
Error

Example

```
var markData = {
  "type": "RectangleAnnotation",
  "rectangle": {
    "x": 0,
    "y": 10,
    "width": 200,
    "height": 200
  },
  "fillColor": "#FB0404",
  "lineWidth": 4,
  "pageNumber": 1,
  "uid":
  "dW9qaV8yMDE5LTA4LTAxVDA2OjUzOjA1LjgxoVpfazhhbDFi",
  "data": {},
  "conversation": {
    "data": {}
  }
};

// Deserialize mark from object value
viewerControl.deserializeMarks(markData);

// Deserialize mark from JSON string
var markJson = JSON.stringify(markData);
viewerControl.deserializeMarks(markJson);

// Deserialize marks from array of objects
var markData2 = { ... };
viewerControl.deserializeMarks([markData, markData2]);
```

destroy()

Closes the `ViewerControl` and cleans up its resources. After this action, a new viewer can be created in its place.

Note: If the viewer was created using the jQuery plugin ([PCCViewer.Viewer](#)), use the `PCCViewer.Viewer.destroy` method instead.

Example

```
var element = document.querySelector('#mydiv');
var viewerControl = new PCCViewer.ViewerControl(element,
options);

viewerControl.destroy();
```

disposePageText(pageNumber) → {[PCCViewer.ViewerControl](#)}

Disposes text for the specified page.

The text for any page requested programmatically by using the [PCCViewer.ViewerControl#requestPageText](#) method is not automatically disposed, even when the `ViewerControl` `discardOutOfViewText` parameter is set to true. You must use the `disposePageText` method to dispose of the text when you no longer need it.

If the specified page is currently displayed when calling this method, the page text will not be disposed immediately, but will be disposed when the page is scrolled out of view and no longer displayed.

Client search depends on the page text, so page text is not disposed after performing client search.

Parameters:

Name	Type	Description
<code>pageNumber</code>	number	Page text is disposed for this page number.

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type
Error

- If the `pageNumber` value is not a number.

Type
Error

- If the `pageNumber` value is out of range.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
viewerControl.disposePageText(10);
```

`documentHasText()` → [{PCCViewer.Promise}](#)

Indicates whether or not any pages in the document have text. This method returns a promise, which can resolve at an indefinite time. This promise will resolve at the first instance of finding text in the document, but will not actively search for text if none is yet found. Unless this promise has resolved, the user should assume that the document does not contain text.

For example, in a 100 page document, where only page 80 has text, this promise will not resolve until the user views page 80. If the document were to have no text at all, the promise will not resolve until all 100 pages were viewed by the user.

The successful callback be passed only a boolean indicating whether the document has any text or not.

Returns:

a Promise object.

Type
[PCCViewer.Promise](#)

Example

```
var promise = viewerControl.documentHasText().then(
    function success(containsTextBool) {
        alert('Document has text: ' + (containsTextBool ?
'Yes' : 'No'));
    },
    function fail(error) {
        alert('Document has text: unknown due to error "'
+ error.message + '"');
    }
);
```

`enterTextMarkEditingMode(mark)` → `{PCCViewer.ViewerControl}`

Puts a displayed `PCCViewer.Mark` object of type `TextInputSignature`, `TextAreaSignature`, `TextRedaction`, or `TextAnnotation` into editing mode. In editing mode, a text input or text box will be drawn for the mark and the input will have focus. When in editing mode, the end user type to modify the text of the mark.

All marks other than the specified mark will be taken out of text mark editing mode.

Note that if the mark is not on a page that is currently displayed, then this method may have no perceived effect. The act of changing pages to bring the off screen page into view would take the focus off of the mark that is in editing mode.

Parameters:

Name	Type	Description
mark	<code>PCCViewer.Mark</code> null undefined	A <code>PCCViewer.Mark</code> object that will be put in editing mode. If a value of <code>null</code> or <code>undefined</code> is given, then all marks will be taken out of text mark editing mode.

Throws:

- If the `PCCViewer.EventType.ViewerReady` event was not fired prior to calling this method.

Type

`PCCViewer.Error`

- If `mark` is not a valid `PCCViewer.Mark` object or the mark provided is not currently on the document.

Type

`PCCViewer.Error`

- If `mark` is not of type `TextInputSignature`, `TextAreaSignature`, `TextRedaction`, or `TextAnnotation`.

Type

`PCCViewer.Error`

Returns:

The `ViewerControl` object on which this method was called.

Type

`PCCViewer.ViewerControl`

Example

```
// Add a mark
var myMark = viewerControl.addMark(4,
PCCViewer.Mark.Type.TextAnnotation)
    .setRectangle({
        x: 0,
        y: 600,
        width: 500,
        height: 100
    });

// Put the mark in editing mode and scroll to the mark.
viewerControl.enterTextMarkEditingMode(myMark);
```

fitContent(fitType) → {[PCCViewer.ViewerControl](#)}

Changes the scaling (zoom) of the document to fit the content in the viewer. How the content is fit in the viewer is based on the specified by the values in the [PCCViewer.FitType](#) enumerable.

Parameters:

Name	Type	Description
fitType	string	Specifies how the content will be scaled to fit in the viewer.

See: [PCCViewer.FitType](#) for a list of possible FitType values and their descriptions.

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.
Type
Error
- If the value of fitType is unknown.
Type
Error
- If the view mode is set to "EqualFitPages" or "EqualWidthPages", and the value of fitType is "ActualSize". Instead, pass a value of 1 to the [PCCViewer.ViewerControl#setScaleFactor](#) method to scale the first page to actual size.
Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type

`PCCViewer.ViewerControl`

Example

```
// Explicitly specify the fit type
viewerControl.fitContent("FullWidth");

// or use the enumeration
viewerControl.fitContent(PCCViewer.FitType.FullWidth);
```

`getActiveMarkupLayer()` → `{PCCViewer.MarkupLayer}`

Gets the viewer control's active markup layer.

Returns:

The viewer control's active markup layer.

Type

`PCCViewer.MarkupLayer`

Example

```
var activeMarkupLayer =
viewerControl.getActiveMarkupLayer();
```

`getAllMarks()` → `{Array.<PCCViewer.Mark>}`

Gets all marks.

Returns:

An array of `PCCViewer.Mark` objects. **Note:** Returns an empty array if the viewer has not been initialized.

Type

`Array.<PCCViewer.Mark>`

Example

```
var allMarks = viewer.getAllMarks();
```


getAtMaxScale() → {boolean}

Gets a value that indicates whether the viewer is currently at the maximum scale factor. As long as this value is `true`, the `ViewerControl` do nothing if asked to zoom in any further.

This method determines the value each time, and will be affected by the following:

1. The viewer scale changes due to `zoomIn`, `zoomOut`, or `fitContent`.
2. The window resizes.
3. The div that holds the viewer control is resized.

See: [PCCViewer.ViewerControl#atMaxScale](#)
[PCCViewer.EventType.ScaleChanged](#)

Throws:

If the [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type
Error

Returns:

A value that indicates whether the viewer is currently at maximum zoom.

Type
boolean

Examples

```
// Check for change after the viewer scale changes
viewerControl.on(PCCViewer.EventType.ScaleChanged,
function(ev) {
    var atMax = viewer.getAtMaxScale();
}
```

```
// Check for change when the window resizes (using
jQuery)
$(window).resize(function() {
    var atMax = viewer.getAtMaxScale();
});
```

getAtMinScale() → {boolean}

Gets a value that indicates whether the viewer is currently at the minimum scale factor. As long as this value is `true`, the `ViewerControl` do nothing if asked to zoom out any further.

This method determines the value each time, and will be affected by the following:

1. The viewer scale changes due to `zoomIn`, `zoomOut`, or `fitContent`.
2. The window resizes.
3. The div that holds the viewer control is resized.

See: [PCCViewer.ViewerControl#atMinScale](#)
[PCCViewer.EventType.ScaleChanged](#)

Throws:

If the [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type
Error

Returns:

A value that indicates whether the viewer is currently at minimum zoom.

Type
boolean

Examples

```
// Check for change after the viewer scale changes
viewerControl.on(PCCViewer.EventType.ScaleChanged,
function(ev) {
    var atMax = viewer.getAtMinScale();
}
```

```
// Check for change when the window resizes (using
jQuery)
$(window).resize(function() {
    var atMax = viewer.getAtMinScale();
});
```

[getCharacterIndex\(sortableObject\)](#) → {Number}

Returns the index of the character at the location of the specified object. If there is no character at the location, the index of the character before the location (based on the closest line of text) is returned.

Parameters:

Name	Type	Description
sortableObject	PCCViewer.Mark PCCViewer.SearchResult Object	A mark or search result in a page, or an object containing <code>pageNumber</code> , <code>x</code> , and <code>y</code> values.

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type
Error

- If sortableObject is not a valid [PCCViewer.Mark](#) or [PCCViewer.SearchResult](#), or if it is not an Object with valid pageNumber, x, and y values. When specifying a pageNumber greater than 1, this method requires that the PageCountReady event has been triggered, otherwise an error is thrown.

Type
Error

Returns:

The text index at or before the position of the specified object.

Type
Number

Example

```
// get the first selected Mark object
var theFirstMark = viewerControl.getSelectedMarks()[0];

// get the sort index
if (theFirstMark) var firstMarkSortIndex =
viewerControl.getCharacterIndex(theFirstMark);
```

[getConversationDOMFactory\(\)](#) → {function}

Gets the conversation DOM factory function. The default factory function is returned if a factory function has not been set using the [PCCViewer.ViewerControl#setConversationDOMFactory](#) method.

Returns:

The function for creating a conversation DOM element.

Type
function

[getCurrentMouseTool\(\)](#) → {string}

Gets the current mouse tool of the viewer.

See: [PCCViewer.MouseTools.getMouseTool](#)

Returns:

A value indicating the name of the current mouse tool.

Type
string

Example

```
// get the current mouse tool name
var mouseToolName = viewerControl.getCurrentMouseTool();

// get the actual MouseTool object
var mouseToolObject =
PCCViewer.MouseTools.getMouseTool(mouseToolName);
```

[getDownloadDocumentURL\(\)](#) → {string}

Gets the URL to download the original document.

See: The help section titled "Digital Rights Management Configuration" for information on disabling document download.

Returns:

The URL for downloading the original document. This URL is relative to current page.

Type
string

Example

```
// get the URL
var documentURL = viewerControl.downloadDocument();

// download the document
window.location.href = documentURL;
```

[getGamma\(\)](#) → {Number}

Gets the gamma factor for the document.

Returns:

The current gamma factor for the document.

Type
Number

Example

```
var gamma = viewerControl.getGamma();
```

`getIsCommentsPanelOpen()` → {boolean}

Returns a value (`true` or `false`) indicating if the comments panel is open.

Throws:

If `PCCViewer.EventType.ViewerReady` event was not fired prior to using this method.

Type
Error

Returns:

A value indicating if the comments panel is open.

Type
boolean

Example

```
var isCommentsPanelOpen =  
viewerControl.getIsCommentsPanelOpen(); // true if  
comments panel is open
```

`getMarkById(markId)` → {PCCViewer.Mark}

Gets the specified mark.

Parameters:

Name	Type	Description
markId	number	The ID of the mark to retrieve.

Returns:

The mark that corresponds to the specified ID.

Type

[PCCViewer.Mark](#)

Example

```
var mark = viewer.getMarkById(1);
```

[getMarkHandleMode\(\)](#) → [{PCCViewer.MarkHandleMode}](#)

Gets the mark handle mode.

Returns:

The mark handle mode.

Type

[PCCViewer.MarkHandleMode](#)

[getMarksByType\(markType\)](#) → [{Array.<PCCViewer.Mark>}](#)

Get marks by type.

Parameters:

Name	Type	Description
markType	PCCViewer.Mark.Type string	The mark type being requested. Note: Returns an empty array if the viewer has not been initialized.

See: [PCCViewer.Mark.Type](#) for a list of valid mark types.

Throws:

If the parameter markType is an invalid mark type.

Type

Error

Returns:

An array of [PCCViewer.Mark](#) objects of the requested type.

Type

Array.<[PCCViewer.Mark](#)>

Example

```
var marksByType = viewer.getMarksByType(markType);
```

[getMarkupLayerCollection\(\)](#) → {[PCCViewer.MarkupLayerCollection](#)}

Gets the viewer control's markup layer collection.

Returns:

The viewer control's markup layer collection.

Type

[PCCViewer.MarkupLayerCollection](#)

Example

```
var markupLayerCollection =  
viewerControl.getMarkupLayerCollection();
```

[getMaxScaleFactor\(\)](#) → {number}

Gets the maximum scale limit.

Throws:

If the [PCCViewer.EventType.ViewerReady](#) event was not fired prior to calling this method.

Type

Error

Returns:

A number indicating the maximum limit at which the document can be scaled.

Type

number

Example

```
var maxScaleFactor = viewerControl.getMaxScaleFactor();  
viewerControl.setScaleFactor(maxScaleFactor); // Zoom in
```

```
to the maximum scale limit.
```

`getMinScaleFactor()` → {number}

Gets the minimum scale limit.

Throws:

If the [PCCViewer.EventType.ViewerReady](#) event was not fired prior to calling this method.

Type
Error

Returns:

A number indicating the minimum limit at which the document can be scaled.

Type
number

Example

```
var minScaleFactor = viewerControl.getMinScaleFactor();  
viewerControl.setScaleFactor(minScaleFactor); // Zoom out  
to the minimum scale limit.
```

`getPageCount()` → {number}

Gets the known page count of the current document.

This value is updated when the viewer gets the page count or estimated page count from the server. Subscribe to the [PCCViewer.EventType.PageCountReady](#) and [PCCViewer.EventType.EstimatedPageCountReady](#) events to be notified when the viewer gets the page count from the server.

The initial value is 1, before any page count event.

See: [PCCViewer.EventType](#) for the event type "PageCountReady".

Returns:

The known page count.

Type
number

Example


```
// First, create the pccViewer.
var viewerControl =
$("#viewer").pccViewer(viewerOptions).viewerControl;

function pageCountReadyHandler(event) {
    // The page count has now been determined.
    var pageCount = viewer.getPageCount();
    alert("Number of pages = " + pageCount);

    // Now, unsubscribe from the event.
    viewerControl.off("PageCountReady",
pageCountReadyHandler);
}

// Subscribe to the PageCountReady event exposed by the
API
viewerControl.on("PageCountReady",
pageCountReadyHandler);
```

getPageLayout() → {[PCCViewer.PageLayout](#)}

Gets the page layout. This defines how the document pages will be arranged, based on the values of the [PCCViewer.PageLayout](#) enumeration.

See: [PCCViewer.PageLayout](#)

Returns:

A value indicating the page layout.

Type

[PCCViewer.PageLayout](#)

Example

```
var pageLayout = viewerControl.getPageLayout();
```

getPageNumber() → {[number](#)}

Gets the current page number of the viewer. This is a 1-based value, so the first page of a document is 1.

See: [PCCViewer.EventType.PageCountReady](#) event

Returns:

The current page of the viewer.

Note: A value of 1 is returned before page count is available.

Type
number

Example

```
var currentPageNumber = viewerControl.getPageNumber();
```

getPageRotation(pageNumber) → {number}

gets the page rotation value for the specified page number.

Parameters:

Name	Type	Description
pageNumber	number string	This is an optional parameter. A 1-based page number of the page. An optional string representation of a valid number is also accepted as a parameter. If this parameter is not provided, the implied pageNumber will be the currentPage.

Throws:

- If the [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type
Error

- If the provided optional parameter `pageNumber` is less than 1 or greater than the value returned by [PCCViewer.ViewerControl#getPageCount](#).

Type
Error

- If the provided optional parameter `pageNumber` is not a number or a string representation of a number.

Type
Error

- If the provided optional parameter `pageNumber` is not an integer page number.

Type
Error

Returns:

The rotation amount in degrees clockwise.

Type
number

Example

```
// first create the pccViewer : Note: if the viewer
object has already been created, do not re-create it.
var viewerControl =
$("#viewer").pccViewer(viewerOptions).viewerControl;

var pageNumber = 2;
var rotationAngle =
viewerControl.getPageRotation(pageNumber);
```

getRedactionViewMode() → {[PCCViewer.RedactionViewMode](#)}

Gets the redaction view mode. This defines whether the redaction rectangles would show the underlying document content text [PCCViewer.RedactionViewMode](#) enumerable values.

See: [PCCViewer.RedactionViewMode](#)

Returns:

A value indicating the redaction view mode.

Type
[PCCViewer.RedactionViewMode](#)

Example

```
var viewMode = viewerControl.getRedactionViewMode();
```

getSavedMarkupNames() → {[PCCViewer.Promise](#)}

Gets a list of all saved markups from the server for the current document.

If unable to retrieve markup list from server, then the returned [PCCViewer.Promise](#) object is rejected with the reason set to a [PCCViewer.Error](#) object with its code property set to `MarkupListFail`.

If AJAX is not supported, then the returned [PCCViewer.Promise](#) object is rejected with the reason set to a [PCCViewer.Error](#) object with its code property set to `AjaxUnsupported`.

If a server error is encountered, then the returned [PCCViewer.Promise](#) object is rejected with the reason set to a [PCCViewer.Error](#) object with its code property set to `Error`.

The [PCCViewer.Promise~onFulfilled](#) function gets passed an `Array` of `Objects`. Each object will have a `name` property, which is a string representation of the name used to save the markup.

Returns:

A [PCCViewer.Promise](#) object.

Type

[PCCViewer.Promise](#)

Example

```
viewerControl.getSavedMarkupNames().then(
  function onSuccess(markupNameObjects) {
    var namesArray = [];
    for (var i = 0; i < markupNames.length; i++) {
      namesArray.push(markupNames[i].name);
    }
    alert(namesArray.join(', '));
  },
  function onFailure(error) {
    alert((error.message ? error.message : error));
  }
);
```

[getScaleFactor\(\)](#) → {number}

Gets the scale factor of the viewer.

See: [PCCViewer.EventType.ScaleChanged](#) event

Returns:

A value indicating the scale factor to use in the viewer, with 1 being 100% zoom.

For "Document" and "SinglePage" view mode, the scale factor represents the amount that each page in the document is scaled. For "EqualFitPages" view mode, the scale factor represents the amount that the first page of the document is scaled.

Type

number

Example

```
var scaleFactor = viewerControl.getScaleFactor();
```

[getSearchRequest\(\)](#) → [{PCCViewer.SearchRequest}](#)

Gets the `SearchRequest` object from the last call to [PCCViewer.ViewerControl#search](#).

Returns:

The `SearchRequest` from the last search, or `null` if a search has not been performed or if the search has been cleared with [PCCViewer.ViewerControl#clearSearch](#).

Type

[PCCViewer.SearchRequest](#)

Example

```
var searchRequestA = viewerControl.search("Foo");
var searchRequestB = viewerControl.getSearchRequest();
searchRequestA === searchRequestB; // true
```

[getSelectedConversation\(\)](#) → [{PCCViewer.Conversation}](#)

Gets the selected conversation.

See:

[PCCViewer.ViewerControl#setSelectedConversation](#)

Returns:

The selected conversation, or `null` if no conversation is currently selected.

Type

[PCCViewer.Conversation](#)

Example

```
var selectedConversation =
viewerControl.getSelectedConversation();
```

[getSelectedMarks\(\)](#) → [{Array.<PCCViewer.Mark>}](#)

Obtains all the selected marks in the currently loaded document. If none of the marks are selected or if the document does not contain any marks then the returned array will be empty.

Returns:

An array of selected [PCCViewer.Mark](#) objects.

Type

Array.<[PCCViewer.Mark](#)>

Example

```
var selectedMarks = viewerControl.getSelectedMarks();

if (selectedMarks.length) alert(selectedMarks.length + "
marks are currently selected");
else alert("No marks are currently selected");
```

[getSelectedSearchResult\(\)](#) → {[PCCViewer.SearchResult](#)|null}

Gets the selected [PCCViewer.SearchResult](#) object.

Returns:

The [PCCViewer.SearchResult](#) object. Returns null if no search result is selected.

Type

[PCCViewer.SearchResult](#) | null

[getSharpening\(\)](#) → {[Number](#)}

gets the sharpening factor for the document.

Returns:

the current sharpening factor for the document.

Type

Number

Example

```
var sharpening = viewerControl.getSharpening();
```

[getSvgLineWidthMultiplier\(\)](#) → {[Number](#)}

Gets the current multiplier used to adjust SVG line widths in the document.

Returns:

The current svg line-width multiplier for the document.

Type
Number

Example

```
var svgLineWidthMultiplier =  
viewerControl.getSvgLineWidthMultiplier();
```

getViewMode() → {[PCCViewer.ViewMode](#)}

Gets the view mode. This defines how the document pages will be scaled, based on the values of the [PCCViewer.ViewMode](#) enumerable values.

See: [PCCViewer.ViewMode](#)

Returns:

A value indicating the view mode.

Type
[PCCViewer.ViewMode](#)

Example

```
var viewMode = viewerControl.getViewMode();
```

hideMarks(marks) → {[PCCViewer.ViewerControl](#)}

Hides the specified marks.

Parameters:

Name	Type	Description
marks	Array.< PCCViewer.Mark >	An Array of objects of type PCCViewer.Mark .

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type
Error

- If any of the marks passed in are not valid objects of the type [PCCViewer.Mark](#).

Type

Error

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)**Example**

```
// hides all selected marks
viewerControl.hideMarks(viewer.getSelectedMarks());
```

`isPageTextReady(pageNumber)` → {boolean}

Returns a value indicating if the `ViewerControl` has loaded text for a page from the server.

Several of the `ViewerControl`'s methods require that the `ViewerControl` has loaded the text for the page. These methods will not work if the text for the target page(s) is not loaded. This set of methods includes:

- [PCCViewer.ViewerControl#getCharacterIndex](#)
- [PCCViewer.Mark#setPosition](#)

The `ViewerControl` may delay loading of text for a page, so it is not safe to assume that the `ViewerControl` has text for a page when `ViewerReady` is triggered. *However, if you have gotten page text through the viewer, then it is safe to assume the viewer has text for the page.* For example, if you get text through a [SearchResult](#) generated by [PCCViewer.ViewerControl#search](#) or by calling [PCCViewer.ViewerControl#requestPageText](#), then the viewer will have loaded page text from the server.

Parameters:

Name	Type	Description
<code>pageNumber</code>	number	The method checks if page text is ready for this page number.

See: [PCCViewer.ViewerControl#requestPageText](#)
[PCCViewer.ViewerControl#search](#)
[PCCViewer.EventType.PageTextReady](#)

Throws:

If the `pageNumber` argument is less than 1 or greater than the page count of the document.

Type

Error

Returns:

A value indicating if the ViewerControl has loaded text for a page from the server.

Type

boolean

Example

```
if (!viewerControl.isPageTextReady(1)) {
    viewerControl.requestPageText(1).then(
        function(pageText) {
            highlightSomeText({
                startIndex: 0,
                length: pageText.length
            });
        }
    );
} else {
    // Note that it's pretty silly to blindly highlight
    // text, you would almost always want to know
    // the text tht you are trying to highlight before
    // calling the API to set the position. But, this is just
    // an example to demonstrate when you don't *have* to
    // request the page text.
    highlightSomeText({startIndex: 0, length: 5});
}

// Note: an alternative to the if-else above, would be to
// always call requestPageText, which will immediately
// resolve
// if the viewer already has page text. Using this
// approach, you don't need to call `isPageTextReady`.
// viewerControl.requestPageText(1).then(
//     function(pageText) {
//         highlightSomeText({
//             startIndex: 0,
//             length: pageText.length
//         });
//     }
// );

function highlightSomeText(position) {
    viewerControl.addMark(1, "HighlightAnnotation")
        .setPosition(position);
}
```

loadAttachments() → {PCCViewer.Promise}

Gets a list of all attachments of the document currently loaded on the viewer.

If unable to retrieve the attachment list from server, then the returned `PCCViewer.Promise` object is rejected with the reason set to a `PCCViewer.Error` object with its code property set to `AttachmentListFail`.

If AJAX is not supported, then the returned `PCCViewer.Promise` object is rejected with the reason set to a `PCCViewer.Error` object with its code property set to `AjaxUnsupported`.

If a server error is encountered, then the returned `PCCViewer.Promise` object is rejected with the reason set to a `PCCViewer.Error` object with its code property set to `Error`.

The `PCCViewer.Promise~onFulfilled` function gets passed an array of attachment objects.

Returns:

A `PCCViewer.Promise` object.

Type

`PCCViewer.Promise`

Example

```
viewerControl.loadAttachments().then(
  function onSuccess(attachments) {
    var fileNames = [];
    for (var i = 0; i < attachments.length; i++) {
      fileNames.push(attachments[i].name);
    }
    alert(fileNames.join(', '));
  },
  function onFailure(error) {
    alert((error.message ? error.message : error));
  }
);
```

loadMarkup(recordName, retainExistingMarks_{opt}, markupLayer_{opt}) → `{PCCViewer.Promise}`

Loads the markup with the specified name from the server. This returns a `PCCViewer.Promise` object.

If the parameter `recordName` is invalid, then the `PCCViewer.Promise` object that is returned will be rejected with the reason set to a `PCCViewer.Error` object with its code property set to `InvalidAnnotationRecord`.

If the optional parameters `retainExistingMarks` or `markupLayer` are specified but invalid, then the `PCCViewer.Promise` object that is returned will be rejected with the reason set to a `'PCCViewer.Error'` object.

If a server error is encountered, then the returned `PCCViewer.Promise` object is rejected with the reason set to a `PCCViewer.Error` object with its code property set to `Error`.

If AJAX is not supported, then the returned `PCCViewer.Promise` object is rejected with the reason set to a `PCCViewer.Error` object with its code property set to `AjaxUnsupported`.

Note: Any existing marks in the document are removed before marks are loaded.

Note: This method triggers the following events: [PCCViewer.EventType.MarkCreated](#), [PCCViewer.EventType.MarkChanged](#) and [PCCViewer.EventType.MarkChanged](#).

Parameters:

Name	Type	Attributes	Default	Description
recordName	string			Name of the annotation record to be loaded.
retainExistingMarks	boolean	<optional>	true	If true, retains the existing marks.
markupLayer	PCCViewer.MarkupLayer	<optional>		If set to a valid markup layer, the marks are added to the specified markup layer. If not set, the marks are added to the currently active markup layer. Otherwise, clears the existing marks before loading new marks.

See: [PCCViewer.ViewerControl#getSavedMarkupNames](#)

Returns:

a Promise object.

Type

[PCCViewer.Promise](#)

Example

```
viewerControl.loadMarkup("mymarkup").then(
  function onResolved() {
    // update the UI, or whatever... because the markup
    loaded successfully
  },
  function onRejected(error) {
    alert("loading failed! " + (error.message ?
    error.message : error));
  });
```

loadMarkupLayers(layerRecordIds) → [{PCCViewer.Promise}](#)

Load markup layer records from the server. This method loads one or more layers from the server asynchronously, and returns a `Promise` to resolve the request.

The `onFulfilled` callback will receive an object containing [PCCViewer.MarkupLayer](#) objects representing the loaded layers.

The `onRejected` callback will receive a `PCCViewer.Error` object defining why the load function failed.

Parameters:

Name	Type	Description
<code>layerRecordIds</code>	<code>string Array.<string></code>	A string or an array of layer record IDs.

Throws:

For markup that contains marks of type `RectangleRedaction` and `TextSelectionRedaction` if the `reasons` property defined for them and `ViewerControlOptions.enableMultipleRedactionReasons` property was set to `false` when initializing `ViewerControl`.

Type
`Error`

Returns:

A `Promise` object.

Type
`PCCViewer.Promise`

Example

```
// load a layer

viewerControl.loadMarkupLayers('abc123').then(
  function onSuccess(layers) {
    console.log('layer saved successfully',
layers[0].getId());
  },
  function onFailure(reason) {
    console.log('layer failed to load:', reason.code,
reason.message);
  }
);
```

`moveMarkBackward(mark)` → `{PCCViewer.ViewerControl}`

Moves the specified mark one slot backward from its position in the internal `z-order`. When the marks overlap, marks that are higher in this internal `z-order` are drawn over the marks that are lower.

Parameters:

Name	Type	Description
mark	PCCViewer.Mark	The mark being moved.

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type
Error

- If the mark is an invalid [PCCViewer.Mark](#) object or a template mark.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
// get the first selected Mark object
var theFirstMark = viewerControl.getSelectedMarks()[0];

// move it backward
if (theFirstMark)
viewerControl.moveMarkBackward(theFirstMark);
```

`moveMarkForward(mark)` → **`{PCCViewer.ViewerControl}`**

Moves the specified mark one slot toward the top of the internal `z-order`. When the marks overlap, marks that are higher in this internal `z-order` are drawn over the marks that are lower.

Parameters:

Name	Type	Description
mark	PCCViewer.Mark	The mark being moved.

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type
Error

- If the mark is an invalid [PCCViewer.Mark](#) object or a template mark.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
// get the first selected Mark object
var theFirstMark = viewerControl.getSelectedMarks()[0];

// move it forward
if (theFirstMark)
viewerControl.moveMarkForward(theFirstMark);
```

`moveMarkToBack(mark)` → **`{PCCViewer.ViewerControl}`**

Moves the specified mark to the back. When the marks overlap, the marks that are higher in the internal `z-order` on a page are drawn over the ones that are lower.

Parameters:

Name	Type	Description
mark	PCCViewer.Mark	The mark being moved.

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event has not fired prior to calling this method.

Type
Error

- If the mark is an invalid [PCCViewer.Mark](#) object or a template mark.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type

`PCCViewer.ViewerControl`

Example

```
// get the first selected Mark object
var theFirstMark = viewerControl.getSelectedMarks()[0];

// move it forward
if (theFirstMark)
viewerControl.moveMarkForward(theFirstMark);
```

`moveMarkToFront(mark)` → `{PCCViewer.ViewerControl}`

Moves the specified mark to the front or to the top of internal `z-order` on a page. When the marks overlap, the marks with higher internal `z-order` are drawn over the ones that are lower.

Parameters:

Name	Type	Description
mark	<code>PCCViewer.Mark</code>	The mark being moved.

Throws:

- If `PCCViewer.EventType.ViewerReady` event was not fired prior to using this method.

Type

Error

- If the mark is an invalid `PCCViewer.Mark` object or a template mark.

Type

Error

Returns:

The `ViewerControl` object on which this method was called.

Type

`PCCViewer.ViewerControl`

Example

```
// get the first selected Mark object
var theFirstMark = viewerControl.getSelectedMarks()[0];

// move it forward
if (theFirstMark)
viewerControl.moveMarkForward(theFirstMark);
```

off(eventType, handler) → {[PCCViewer.ViewerControl](#)}

Unsubscribe an event handler from a specified event type.

Typically, an event is unsubscribed when you no longer want further notification of the event.

Parameters:

Name	Type	Description
eventType	string	A string specifying the event type. See PCCViewer.EventType for a list and description of all supported events.
handler	PCCViewer.Event~eventHandler	A function that was attached previously to the <code>ViewerControl</code> . Note: This must be the same function object previously passed to PCCViewer.ViewerControl#on . It cannot be an different object that is functionally equivalent.

See: [PCCViewer.ViewerControl#on](#)

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)

Example

```
// Our event handler declaration
function handler(event) {
    alert("An event was fired: " + event.getType());
}

// Subscribe
viewerControl
    .on("PageChanged", handler)
    .on("PageCountReady", handler);
```



```
// Un-subscribe
viewerControl
    .off("PageChanged", handler)
// Use string literals or the enum.
    .off(PCCViewer.EventType.PageCountReady, handler);
// Chain unsubscription.
```

on(eventType, handler) → {PCCViewer.ViewerControl}

Subscribe an event handler to an event of a specified type.

Parameters:

Name	Type	Description
eventType	string	A string that specifies the event type. This value is case-insensitive. See PCCViewer.EventType for a list and description of all supported events.
handler	PCCViewer.Event~eventHandler	A function that will be called whenever the event is triggered.

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)

Example

```
// Create the viewer and get the ViewerControl object.
var viewerControl =
    $("#viewer").pccViewer(viewerOptions).viewerControl;

// Our event handler declaration
function handler(event) {
    alert("An event was fired: " + event.getType());
}

viewerControl
    .on(PCCViewer.EventType.PageChanged, handler) //
// Use the PCCViewer.EventType enum.
    .on(PCCViewer.EventType.PageDisplayed, handler) //
// Chain event subscription.
    .on("PageLoadFailed", handler); //
// Use string literals instead of the EventType enum.
```

`openCommentsPanel()` → `{PCCViewer.ViewerControl}`

Opens the comments panel.

Throws:

- If `PCCViewer.EventType.ViewerReady` event was not fired prior to using this method.

Type
Error

- If the page layout is set to "Horizontal".

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
`PCCViewer.ViewerControl`

Example

```
viewerControl.openCommentsPanel();
```

`print(optionsopt)` → `{PCCViewer.PrintRequest}`

Print the document associated with the `PCCViewer.ViewerControl` object. Note, the "print" method will only work if the correct "printTemplate" value is passed to "ViewerControlOptions" while initializing the "ViewerControl" instance. It is recommended that this value be set to the content of the "printTemplate.html" file. If this value is set to an empty string (default), then printing will be unavailable.

Parameters:

Name	Type	Attributes	Description		
options	Object	<optional>	Provides instructions for what to print. The object may have the following		
Properties					
Name	Type	Attributes	Default	Description	
range	string	<optional>	"all"	A string representing the range of pages to print. The range is separated by hyphens and commas. For example, "1-10, 15-20" prints pages 1 through 10, 15 through 20, and all other pages in the document.	
orientation	string	<optional>	"portrait"	Describes the orientation of the page. Possible values are "portrait" and "landscape".	
paperSize	string	<optional>	"letter"	Describes which paper size should be printed. Possible values are "letter", "a4", and "a5".	
margins	string	<optional>	"default"	Whether to print margins. This property is only applicable when printing a range of pages. Possible values are "default", "none", and "all".	
includeMarks	boolean	<optional>	false	Whether to print annotations. Possible values are "true" and "false". Note: PCCV returns true by default.	
includeAnnotations	boolean	<optional>	false	Whether to print annotations. Possible values are "true" and "false". Note: PCCV returns true by default.	

Name	Type	Attributes	Description
			<p>includeRedactions boolean <optional> false</p> <p>Whether to p</p> <ul style="list-style-type: none"> • Possib • Note: PCCV return
			<p>includeComments string <optional> "none"</p> <p>Location to p</p> <ul style="list-style-type: none"> • Possib "fol "doc
			<p>includeReasons string <optional> "none"</p> <p>Location to p</p> <ul style="list-style-type: none"> • Possib "fol "doc
			<p>redactionViewMode string <optional> "Normal"</p> <p>Whether to p underneath s selection tex</p> <ul style="list-style-type: none"> • Possib <ul style="list-style-type: none"> o o

See: Use [PCCViewer.ViewerControl#validatePrintRange](#) to validate a user supplied print range before calling `print`.

Use [PCCViewer.ViewerControl#canPrintMarks](#) to determine if the browser support printing annotations and redactions.

Throws:

If the page(s) are out of range.

Type
Error

Returns:

Type

[PCCViewer.PrintRequest](#)

Example

```
// Prints pages 1 and 3 of the document.  
// Any annotations on those pages will also be printed if  
// supported by the browser.  
viewerControl.print({  
  range : "1, 3",  
  includeMarks : true  
});
```

[refreshConversations\(conversations_{opt}\)](#) → **[{PCCViewer.ViewerControl}](#)**

Forces a DOM refresh of a specified conversation or set of conversations, or all conversations known to the `ViewerControl`.

Parameters:

Name	Type	Attributes	Description
conversations	PCCViewer.Conversation Array. < PCCViewer.Conversation >	<optional>	A single <code>PCCViewer.Conversation</code> object, or an Array of conversation objects. If this parameter is left undefined, all conversations will be refreshed.

See:

[PCCViewer.ViewerControl#setConversationDOMFactory](#)

Throws:

- If the `conversations` parameter is not undefined, a `PCCViewer.Conversation` object, or an array of conversation objects.

Type

Error

- If any of the `PCCViewer.Conversation` objects in the parameter are not conversations of known marks.

Type

Error

- If during the refresh, any call to the DOM Factory does not return a valid `HTMLElement` or a falsy value.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

`requestDocumentConversion(optionsopt)` → `{PCCViewer.ConversionRequest}`

Converts the document and provides an array of URLs to download each converted document. Note that this method currently only supports converting to a single PDF file, so the output array will only contain a single URL.

Parameters:

Name	Type	Attributes	Description								
options	Object	<optional>	<p>This optional parameter specifies conversion options to be used for converting the format of the document.</p> <p>Properties</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Attributes</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>filename</td> <td>string</td> <td><optional></td> <td>Sets the value of the Content-Disposition filename in the header of the response from the URL to download the document. If not set, then the converted document will be downloaded with a default filename.</td> </tr> </tbody> </table>	Name	Type	Attributes	Description	filename	string	<optional>	Sets the value of the Content-Disposition filename in the header of the response from the URL to download the document. If not set, then the converted document will be downloaded with a default filename.
Name	Type	Attributes	Description								
filename	string	<optional>	Sets the value of the Content-Disposition filename in the header of the response from the URL to download the document. If not set, then the converted document will be downloaded with a default filename.								

See: [PCCViewer.ConversionRequest](#) for more details on interacting with the conversion process.

Returns:

A result `ConversionRequest` for this task.

Type
[PCCViewer.ConversionRequest](#)

Example

```
function onSuccessfullConversion(convertedurl) {
    alert("convertedURL = " + convertedurl);
    console.log(convertedurl);
}
function onFailedConversion(error) {
    alert("conversion process failed, error:" +
(error.message ? error.message : error));
}

// A ConversionRequest object is created by and returned
from the call to the convertDocument method
var conversionRequest = viewerControl.convertDocument();
conversionRequest.then(onSuccessfulConversion,
onFailedConversion);

// Register some events
conversionRequest

.on(PCCViewer.ConversionRequest.EventType.ConversionComplete

    function(ev) {
        alert("Document conversion completed.");
    })

.on(PCCViewer.ConversionRequest.EventType.ConversionProgress

    function(event) {
        alert("Conversion progress: " + event.percent +
"%");
    })

.on(PCCViewer.ConversionRequest.EventType.ConversionFailed,

    function(event) {
        alert("Document conversion failed.");
    });

// Also, methods on the conversionRequest object can be
used

// Get the options used to convert the document.
var optionsUsed = conversionRequest.getOptions();

// If the process is still incomplete, cancel can be
used to stop queries to the server.
if(conversionRequest.getProgress() >= 0 &&
conversionRequest.getProgress() < 100) {
    conversionRequest.cancel();
}
```



`requestDocumentHyperlinks(pageNumber)`

Requests the DocumentHyperlinks of the specified page.

The DocumentHyperlinks will be made available via the returned `PCCViewer.Promise` object. The `PCCViewer.Promise~onFulfilled` function will be called with an array of DocumentHyperlinks that are contained on the page. If there are no DocumentHyperlinks on the page, then the returned array will be empty.

If the `pageNumber` parameter is invalid at the time the method is called, or the page does not exist in the document, then the returned `PCCViewer.Promise` object is rejected with the reason set to a `PCCViewer.Error` object with its `code` property set to `Error`.

Parameters:

Name	Type	Description
<code>pageNumber</code>	<code>number</code>	DocumentHyperlinks are requested for this page number.

Example

```
// use
PCCViewer.ViewerControl#requestDocumentHyperlinks (pageNumber)

var promise =
viewerControl.requestDocumentHyperlinks(10);

promise.then(
  function (documentHyperlinks) {
    // Do something with the array of
    DocumentHyperlink objects.
    // In this example we iterate over.
    documentHyperlinks.forEach(function(dh) {
      // Alert the href of each hyperlink.
      // Note: don't actually do that because it
      could be really annoying!
      alert("Found another DocumentHyperlink " +
dh.getHref());
    });
  },
  function (error) {
    alert("Something went wrong " + (error.message ?
error.message : error));
  }
);
```

`requestMarkupLayerNames(metadata)` → `{PCCViewer.Promise}`

Gets a list of all saved markups layer records from the server for the current document. This method utilizes an asynchronous server request to fetch the data, and returns a `Promise` to resolve the request.

The `onFulfilled` callback will receive an object representing the `annotationLayerRecords` persisted on the server.

The `onRejected` callback will receive a `PCCViewer.Error` object defining why the save function failed.

The `PCCViewer.Promise~onFulfilled` function gets passed an `Array of Objects`. Each object will have 1) a `name` property, which is a string representation of the name used to save the markup layer record and 2) a `layerRecordId` property which is the string representation the uniquely identifies the record on the server.

Parameters:

Name	Type	Description
metadata	Object	An optional JSON object that will be passed to the web tier where it can be used for tasks like user identification. The JSON object will be transformed and then passed as GET parameter(s). The metadata JSON object must be flat and only contain values that are strings, numbers, or booleans.

Returns:

A `PCCViewer.Promise` object.

Type

`PCCViewer.Promise`

Example

```
viewerControl.requestMarkupLayerNames().then(
  function onSuccess(annotationLayerRecords) {
    var annotationLayerRecordsArray = [];
    for (var i = 0; i <
annotationLayerRecords.length; i++) {
annotationLayerRecordsArray.push(annotationLayerRecords[i].r
    }
    alert(annotationLayerRecordsArray.join(', '));
  },
  function onFailure(error) {
    alert(error.message ? error.message : error);
  }
);
```

`requestPageAttributes()` → `{PCCViewer.Promise}`

Requests attributes for the specified page.

If the `pageNumber` parameter is invalid at the time the method is called, or the page does not exist in the document, then the returned `PCCViewer.Promise` object is rejected with the reason set to a `PCCViewer.Error` object with its code property set to `RequestPageAttributesFailed`.

If a server error is encountered, then the returned `PCCViewer.Promise` object is rejected with the reason set to a `PCCViewer.Error` object with its code property set to `Error`.

The `PCCViewer.Promise~onFulfilled` function will receive an `Object` with properties `width` and `height`, representing the reported width and height of each page.

Returns:

a `Promise` object.

Type

`PCCViewer.Promise`

Example

```
var promise =
viewerControl.requestPageAttributes(10).then(
    function(pageAttributes) {
        alert('Page 10 attributes: width: ' +
pageAttributes.width + ', height: ' +
pageAttributes.height);
    },
    function(error) {
        alert('Page attributes retrieval for page 10
failed: ' + (error.message ? error.message : error));
    }
);
```

`requestPageText(pageNumber)` → **`{PCCViewer.Promise}`**

Requests the specified text page.

If the `pageNumber` parameter is invalid at the time the method is called, or the page does not exist in the document, then the returned `PCCViewer.Promise` object is rejected with the reason set to a `PCCViewer.Error` object with its code property set to `TextExtractionFailed`.

If a server error is encountered, then the returned `PCCViewer.Promise` object is rejected with the reason set to a `PCCViewer.Error` object with its code property set to `Error`.

The `PCCViewer.Promise~onFulfilled` function will receive a `string` value, representing the text found on that page. If the page has no text, this will be an empty string.

Parameters:

Name	Type	Description
<code>pageNumber</code>	<code>number</code>	Page text is requested for this page number.

Returns:

a Promise object.

Type

[PCCViewer.Promise](#)

Example

```
var promise = viewerControl.requestPageText(10).then(
  function(pageText) {
    alert('Text from page 10: ' + pageText);
  },
  function(error) {
    alert('Text retrieval for page 10 failed: ' +
      (error.message ? error.message : error));
  }
);
```

requestRevisions() → [{PCCViewer.RevisionsRequest}](#)

Requests the revisions for a given document comparison.

Revisions request completes asynchronously. The returned [PCCViewer.RevisionsRequest](#) object provides events for revisions retrieval progress and members to access retrieved revisions.

See: [PCCViewer.RevisionsRequest](#)
[PCCViewer.Revision](#)

Returns:

An object that represents the revisions request. This object allows the calling code to subscribe to revisions retrieval progress events and access retrieved revisions.

Type

[PCCViewer.RevisionsRequest](#)

Example

```
// Request revisions with default options:
var revisionsRequest = viewerControl.requestRevisions();

// Subscribe to the PartialRevisionsAvailable event to
// get revisions as they become available.
revisionsRequest.on('PartialRevisionsAvailable',
function(_event) {
  // Get the newly available revisions:
  var newRevisions = _event.partialRevisions;
```

```
});
```

`rotateDocument(degreesClockwise)` → `{PCCViewer.ViewerControl}`

Rotates all pages in the document by the specified degrees clockwise, relative to each page's current orientation.

Parameters:

Name	Type	Description
<code>degreesClockwise</code>	number	Degrees clockwise to rotate each page. Valid values are multiples of 90: ..., -270, -180, -90, 0, 90, 180, 270, ...

Throws:

- If value of `degreesClockwise` is not valid. The `Error` object will contain property message with details of the error.
- If the `PCCViewer.EventType.ViewerReady` event was not fired prior to using this method.

Type
`Error`

Type
`Error`

Returns:

The `ViewerControl` object on which this method was called.

Type
`PCCViewer.ViewerControl`

Example

```
viewerControl.rotateDocument(90); // Rotates 90
degrees clockwise
viewerControl.rotateDocument(-90); // Rotates 90
degrees counter-clockwise
viewerControl.rotateDocument(180); // Rotates 180
degrees
viewerControl.rotateDocument(540); // Also, rotates 180
degrees
viewerControl.rotateDocument(100); // Throws!
```

`rotatePage(degreesClockwise)` → `{PCCViewer.ViewerControl}`

Rotates the current page by the specified degrees clockwise, relative to the page's current orientation.

Parameters:

Name	Type	Description
<code>degreesClockwise</code>	number	Degrees clockwise to rotate the page. Valid values are multiples of 90: ..., -270, -180, -90, 0, 90, 180, 270, ...

Throws:

- If value of `degreesClockwise` is not valid. The `Error` object will contain property `message` with details of the error.

Type
`Error`

- If the `PCCViewer.EventType.ViewerReady` event was not fired prior to using this method.

Type
`Error`

Returns:

The `ViewerControl` object on which this method was called.

Type
`PCCViewer.ViewerControl`

Example

```
viewerControl.rotatePage(90); // Rotates 90 degrees clockwise
viewerControl.rotatePage(-90); // Rotates 90 degrees counter-clockwise
viewerControl.rotatePage(180); // Rotates 180 degrees
viewerControl.rotatePage(540); // Also, rotates 180 degrees
viewerControl.rotatePage(100); // Throws!
```

`saveMarkup(recordName, optionsopt)` → `{PCCViewer.Promise}`

Saves all markups in the document to the server as a record with a specified name.

If the parameter `recordName` is not a string, then the `PCCViewer.Promise` object that is returned will be rejected with the reason set to a `PCCViewer.Error` object with its code property set to `InvalidArgument`.

If the parameter `recordName` contains invalid characters in the name, then the `PCCViewer.Promise` object that is returned will be rejected with the reason set to a `PCCViewer.Error` object with its code property set to `InvalidCharactersFilename`.

If a server error is encountered, then the returned `PCCViewer.Promise` object is rejected with the reason set to a `PCCViewer.Error` object with its code property set to `Error`.

If AJAX is not supported, then the returned `PCCViewer.Promise` object is rejected with the reason set to a `PCCViewer.Error` object with its code property set to `AjaxUnsupported`.

The `PCCViewer.Promise~onFulfilled` function will receive the `recordName` string as its only argument.

Note: This method will overwrite any previous markup stored with the same name. To prevent duplicates, check to see if the name already exists by using `PCCViewer.ViewerControl#getSavedMarkupNames`.

Parameters:

Name	Type	Attributes	Description
<code>recordName</code>	string		Name of the annotation record saved to an XML file using includes a unique ID for the viewed and the provided annotation name, so valid filename characters are required. When later viewing the saved markup can be located by the record name to the <code>PCCViewer.ViewerControl#loadMarkup</code> method.
<code>options</code>	<code>PCCViewer.ViewerControl~SaveMarkupOptions</code>	<optional>	This optional parameter specifies the markup types to be saved. See <code>PCCViewer.ViewerControl~SaveMarkupOptions</code> details the options object.

Throws:

If `ViewerControlOptions.enableMultipleRedactionReasons` property was set to `true` when initializing `ViewerControl`.

Type
Error

Returns:

a `Promise` object, on success returns the given record name after the record is saved.

Type

PCCViewer.Promise

Example

```
// The following example will save all annotations and
redaction mark types. It will not save signature mark
types.
viewerControl.saveMarkup('cool name').then(
  function success(name) {
    alert('Markup was saved as "' + name + '"');
  },
  function fail(error) {
    alert('Markup was not saved. ' + (error.message ?
error.message : error));
  }
);

// The following will save signature mark types only.
viewerControl.saveMarkup('cool name', {
  includeAnnotations: false,
  includeRedactions: false,
  includeSignatures: true
}).then(
  function success(name) {
    alert('Markup was saved as "' + name + '"');
  },
  function fail(error) {
    alert('Markup was not saved. ' + (error.message ?
error.message : error));
  }
);
```

saveMarkupLayer(id) → {PCCViewer.Promise}

Save a markup layer record to the server. This method saves a single layer to the server asynchronously, and returns a `Promise` to resolve the request.

The `onFulfilled` callback will receive an object containing the `layerRecordId` of the saved layer.

The `onRejected` callback will receive a `PCCViewer.Error` object defining why the save function failed.

Parameters:

Name	Type	Description
<code>id</code>	String	The ID of the layer, as returned by <code>layer.getId()</code> .

Returns:

A `Promise` object.

Type

[PCCViewer.Promise](#)

Example

```
// save the current active layer
var layer = viewerControl.getActiveMarkupLayer();

viewerControl.saveMarkupLayer(layer.getId()).then(
    function onSuccess(layerInfo){
        console.log('layer saved successfully',
layerInfo.layerRecordId);
    },
    function onFailure(reason){
        console.log('layer failed to save:', reason.code,
reason.message);
    }
);
```

scrollBy(offsetX, offsetY) → [{PCCViewer.ViewerControl}](#)

Scrolls by specified offset.

Parameters:

Name	Type	Description
offsetX	number	integer that contains value to scroll horizontally. Negative value will scroll to the left.
offsetY	number	integer that contains value to scroll vertically. Negative value will scroll to the top.

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type

Error

- If target is not an Object with valid offsetX and offsetY values.

Type

Error

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)

Example

```
// scroll by specified offset  
viewerControl.scrollTo(100, 100);
```

scrollTo(target) → {[PCCViewer.ViewerControl](#)}

Scrolls to the specified object.

Parameters:

Name	Type	Description
target	PCCViewer.Mark PCCViewer.Conversation PCCViewer.SearchResult Object	The mark, conversation, search result, or an object (that contains pageNumber, x, and y values) to scroll to.

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type

Error

- If target is not a valid [PCCViewer.Mark](#), [PCCViewer.Conversation](#), or [PCCViewer.SearchResult](#), or if it is not an Object with valid pageNumber, x, and y values.

Type

Error

- If target is a [PCCViewer.Mark](#), [PCCViewer.Conversation](#), or [PCCViewer.SearchResult](#) unknown to the viewer control. When specifying a pageNumber greater than 1, this method requires that the PageCountReady event has been triggered, otherwise an error is thrown.

Type

Error

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)

Example

```
// get the first selected Mark object
var theFirstMark = viewerControl.getSelectedMarks()[0];

// scroll to the mark
viewerControl.scrollTo(theFirstMark);
```

`scrollToAsync(target)` → `{PCCViewer.Promise}`

Scrolls to the specified object.

The returned `PCCViewer.Promise` object will resolve if the page list has completed all scrolling and the page containing the target is displayed.

The returned `PCCViewer.Promise` object is rejected with the reason set to a `PCCViewer.Error` object if:

- The specified target is not a valid `PCCViewer.Mark`, `PCCViewer.Conversation`, or `PCCViewer.SearchResult`, or if it is not an Object with valid `pageNumber`, `x`, and `y` values. When specifying a `pageNumber` greater than 1, this method requires that the `PageCountReady` event has been triggered, otherwise the promise is rejected.
- The specified target is a `PCCViewer.Mark`, `PCCViewer.Conversation`, or `PCCViewer.SearchResult` unknown to the viewer control.
- The `PCCViewer.ViewerControl#scrollTo` or `PCCViewer.ViewerControl#scrollToAsync` method is called before the promise is resolved.

Parameters:

Name	Type	Description
target	<code>PCCViewer.Mark</code> <code>PCCViewer.Conversation</code> <code>PCCViewer.SearchResult</code> Object	The mark, conversation, search result, or an object (that contains <code>pageNumber</code> , <code>x</code> , and <code>y</code> values) to scroll to.

Returns:

A Promise object.

Type

`PCCViewer.Promise`

Example

```
// get the first selected Mark object
var theFirstMark = viewerControl.getSelectedMarks()[0];

// scroll to the mark
var promise =
viewerControl.scrollToAsync(theFirstMark).then(
```

```
function() {
    // after the mark has been scrolled to, select
the mark
    viewerControl.selectMarks([theFirstMark]);
},
function(error) {
    alert('Scrolling failed: ' + (error.message ?
error.message : error));
}
);
```

search(searchQuery) → {PCCViewer.SearchRequest}

Searches the text of the document for the given searchQuery.

This query can be a single search term or a hash specifying one or more terms and options. If only a single search term (string) is supplied, then default options are used.

Search completes asynchronously. The returned [PCCViewer.SearchRequest](#) object, provides events for search progress and members to access search results.

Parameters:

Name	Type	Description
searchQuery	string PCCViewer.ViewerControl~SearchQuery	A value specifying the search query. The value specifies a single search term (string) or an object specifying multiple search terms and options.

See: [PCCViewer.SearchRequest](#)
[PCCViewer.SearchResult](#)

Returns:

An object that represents the search request. This object allows the calling code to subscribe to search progress events and access search results.

Type

[PCCViewer.SearchRequest](#)

Examples

```
// Search on a single term with default options
var searchRequest = viewerControl.search("Hello");

// Subscribe to the PartialSearchResultsAvailable event
to get search results as they become available.
searchRequest.on('PartialSearchResultsAvailable',
function(_event) {
```

```
        // Get the newly available search results.
        var newResults = _event.partialSearchResults;
    });

// Search on multiple terms and specify options
var searchQuery = {
    searchTerms: [{
        searchTerm: "sub",
        contextPadding: 25,
        highlightColor: '#B22222',
        matchingOptions: {
            beginsWith: true,
        }
    },
    {
        searchTerm: "Accusoft"
    }
    ]
};

var requestObject = viewerControl.search(searchQuery);

//subscribe to the search request
requestObject.on('PartialSearchResultsAvailable',
function(_event) {
    var newResults = [];
    //Retrieve results
    newResults = _event.partialSearchResults;
});
```

selectEditorText() → {[PCCViewer.ViewerControl](#)}

Selects the editable text in a mark's text editor

Throws:

- If there is no active text mark editor.
Type
Error
- If the active mark is not a TextInputSignature Mark.
Type
Error
- If the [PCCViewer.EventType.ViewerReady](#) event has not fired prior to calling this method.
Type

Error

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)

Example

```
var viewerControl =
viewerControl.enterTextMarkEditingMode(textAnnotation).select
```

`selectMarks(marks)` → `{PCCViewer.ViewerControl}`

Selects the marks provided in the `Array` parameter.

Note: This method ignores any `Mark` objects with the `interaction mode` set to `PCCViewer.Mark.InteractionMode.SelectionDisabled`.

Parameters:

Name	Type	Description
marks	Array. < PCCViewer.Mark >	An array of PCCViewer.Mark objects that exist in the document and need to be selected.

Throws:

- If `PCCViewer.EventType.ViewerReady` event was not fired prior to calling this method.

Type

Error

- If any of the mark objects are not valid objects, the id of the mark provided does not match the id of mark loaded in the viewer, or the mark provided was not previously added to the document pages.

Type

Error

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)**Example**

```
// get all of the marks
var allMarks = viewerControl.getAllMarks();

// select all of them
viewerControl.selectMarks(allMarks);
```

serializeMarks(marks) → {[PCCViewer.Promise](#)}

Serializes any amount of [Marks](#) passed into the method.

This serialization needs to happen asynchronously, so the method will return a [Promise](#), which will resolve once all marks have been serialized, or get rejected if any error occurs in the process.

A successfully resolved Promise will receive an Array as its only argument, which will contain a plain Object representation of each mark that was passed into the method. These objects can be used with [PCCViewer.ViewerControl#deserializeMarks](#) in order to recreate the same marks at a later time.

Parameters:

Name	Type	Description
marks	PCCViewer.Mark Array.<PCCViewer.Mark>	A single mark or array of marks to be serialized.

See: [PCCViewer.ViewerControl#deserializeMarks](#)

Returns:

A promise object.

Type

[PCCViewer.Promise](#)**Example**

```
// get all selected marks, so we can serialize then
var marksToUse = viewerControl.getSelectedMarks();

viewerControl.serializeMarks(marksToUse).then(
  function success(markObjects) {
    // markObjects is an array of plain objects
    // they can be converted to a JSON string
    var markStr = JSON.stringify(markObjects);
  },
  function fail(reason) {
```

```
        alert('could not serialize marks: ' + reason);
    }
};
```

`serverSearch(searchQuery)` → `{PCCViewer.SearchRequest}`

Searches the text of the document for the given `searchQuery`. The search is performed server-side. This is efficient for larger documents, but for smaller documents it is more efficient to use the `PCCViewer.ViewerControl#clientSearch` method instead.

This query can be a single search term or a hash specifying one or more terms and options. If only a single search term (string) is supplied, then default options are used.

Search completes asynchronously. The returned `PCCViewer.SearchRequest` object, provides events for search progress and members to access search results.

Parameters:

Name	Type	Description
<code>searchQuery</code>	<code>string</code> <code>PCCViewer.ViewerControl~SearchQuery</code>	A value specifying the search query. The value specifies a single search term (string) or an object specifying multiple search terms and options.

See: [PCCViewer.SearchRequest](#)
[PCCViewer.SearchResult](#)

Returns:

An object that represents the search request. This object allows the calling code to subscribe to search progress events and access search results.

Type
`PCCViewer.SearchRequest`

Examples

```
// Search on a single term with default options
var searchRequest = viewerControl.serverSearch("Hello");

// Subscribe to the PartialSearchResultsAvailable event
// to get search results as they become available.
searchRequest.on('PartialSearchResultsAvailable',
function(_event) {
    // Get the newly available search results.
    var newResults = _event.partialSearchResults;
});

// Search on multiple terms and specify options
```

```
var searchQuery = {
  searchTerms: [{
    searchTerm: "sub",
    contextPadding: 25,
    highlightColor: '#B22222',
    matchingOptions: {
      beginsWith: true,
    }
  },
  {
    searchTerm: "Accusoft"
  }
];

var requestObject =
viewerControl.serverSearch(searchQuery);

//subscribe to the search request
requestObject.on('PartialSearchResultsAvailable',
function(_event) {
  var newResults = [];
  //Retrieve results
  newResults = _event.partialSearchResults;
});
```

setActiveMarkupLayer(A)

Sets the viewer control's active markup layer.

Parameters:

Name	Type	Description
A	PCCViewer.MarkupLayer	markup layer.

Throws:

- If markupLayer is not a valid [PCCViewer.MarkupLayer](#).
Type
Error
- If markupLayer is a [PCCViewer.MarkupLayer](#) unknown to the viewer control.
Type
Error

Example


```
viewerControl.loadMarkupLayers('abc123').then(
  function onSuccess(layers) {
    var activeMarkupLayer =
viewerControl.setActiveMarkupLayer(layers[0]);
  },
  function onFailure(reason) {
    console.log('layer failed to load:', reason.code,
reason.message);
  }
);
```

setConversationDOMFactory(factoryFunction) → {PCCViewer.ViewerControl}

Sets the conversation DOM factory function.

Parameters:

Name	Type	Description
factoryFunction	function	<p>This function returns a DOM element that is injected in the viewer control UI, which should show information about the conversation. The viewer control will use this function to obtain a new conversation DOM element whenever:</p> <ul style="list-style-type: none"> • a comment is added • a comment is removed • a comment is modified • a conversation is selected • a conversation is deselected • the PCCViewer.ViewerControl#refreshConversations method is called <p>If this function returns a falsy value, then the conversation will not be shown in the viewer control UI. If this function returns a value that is not a DOM element and not falsy, then the viewer control will throw an exception when attempting to create a conversation DOM element.</p> <p>This function has three parameters, in this order:</p> <ul style="list-style-type: none"> • <code>conversation</code> {PCCViewer.Conversation} - The <code>Conversation</code> in which to generate a DOM element. • <code>state</code> {Object} - An object that indicates the state of the conversation. This object has an <code>isSelected</code> boolean property. • <code>existingDOM</code> {<code>HTMLElement</code> <code>undefined</code>} - The DOM Element returned from the last call to this factory for the given conversation. It is acceptable to return this same element from the factory function if the DOM does not need to be replaced (for example, if it only needs modification or does not require any modification). This parameter may have a value of <code>undefined</code> if the factory function has never been called for this conversation or the previous call did not return a DOM element.

See: [PCCViewer.ViewerControl#setSelectedConversation](#)

Throws:

If the `factoryFunction` parameter is not a `Function`.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
var conversationFactoryFunction = function(conversation,
state, existingDOM) {
    var conversationDiv = document.createElement('div');
    // Create a conversation DOM element.

    // Set the style of the conversation DOM element.
    conversationDiv.style.position = 'absolute';
    if (state.isSelected) { // Set a different background
if the conversation is selected.
        conversationDiv.style.backgroundColor = 'white';
    } else {
        conversationDiv.style.backgroundColor =
'lightgray';
    }
    conversationDiv.style.left = '10px';
    conversationDiv.style.right = '0';
    conversationDiv.style.textAlign = 'left';

    // For each comment in the conversation, create a
comment DOM element.
    var comments = conversation.getComments();
    for (var i = 0; i < comments.length; i++) {
        var commentDiv = document.createElement('div');

        // Set the style of the comment DOM element.
        commentDiv.style.position = 'relative';
        commentDiv.style.padding = '5px';
        commentDiv.style.border = '1px solid gray';
        if (i > 0) {
            commentDiv.style.borderTop = 'none';
```

```
    }

    commentDiv.appendChild(document.createTextNode(comments[i].c
// Add the comment text to the comment DOM element.
        conversationDiv.appendChild(commentDiv); // Add
the comment DOM element to the conversation DOM element.
    }

    return conversationDiv; // Return the conversation
DOM element.
};

viewerControl.setConversationDOMFactory(conversationFactoryE
```

setCurrentMouseTool(name) → {[PCCViewer.ViewerControl](#)}

Sets the current mouse tool of the viewer by name.

See the help section titled "Custom Mouse Tools" for a list of the existing default mouse tools in the viewer.

Parameters:

Name	Type	Description
name	string	The name used when creating the mouse tool. This value is case-insensitive.

See: [PCCViewer.MouseTools.createMouseTool](#)

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event has not fired prior to calling this method.

Type
Error

- If the parameter name is invalid or is an unknown mouseTool type, see, [PCCViewer.MouseTool.Type](#).

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)

setGamma(gammaFactor) → [{PCCViewer.ViewerControl}](#)

Sets the gamma factor for the document. Values from 1 to 10 will darken the document. Values from 0 to 1 will effectively lighten it. Note: This setting only affects viewing and is not persisted when printing or burning.

Parameters:

Name	Type	Description
gammaFactor	number	The gamma factor to set the Gamma for the document to.

Throws:

If the gamma factor is not a valid number between 0 and 10 inclusive.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
viewerControl.setGamma(5); // Darkens content by a factor of 5
viewerControl.setGamma(0.5) // Lightens content by a factor of 0.5
```

setMarkHandleMode(markHandleMode) → [{PCCViewer.ViewerControl}](#)

Sets the mark handle mode.

Parameters:

Name	Type	Description
markHandleMode	PCCViewer.MarkHandleMode	Display mark handles using this mode.

See: [PCCViewer.ViewerControl#setMarkHandleMode](#)

Throws:

If the value of `markHandleMode` is unknown.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
viewerControl.setMarkHandleMode(PCCViewer.MarkHandleMode.Hid
```

[setLayout\(pageLayout\)](#) → [{PCCViewer.ViewerControl}](#)

Sets the layout of the pages.

This property defines the placement or arrangement of the pages in the viewer. The default page layout is "Vertical", in which the pages are displayed as a single vertical column and a vertical scroll bar is displayed to bring into view the pages that are not in view. The "Horizontal" option displays the pages as a single horizontal row and has a horizontal scroll bar to bring into view the pages that are not in the view.

Comments are only supported when using vertical page layout. If the comments panel is open, setting the page layout to horizontal will close the comments panel.

Parameters:

Name	Type	Description
pageLayout	PCCViewer.PageLayout	Display pages using this layout.

See: [PCCViewer.PageLayout](#)

Throws:

If the `pageLayout` value is unknown.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)

Example

```
viewerControl.setPageLayout(PCCViewer.PageLayout.Horizontal)
```

`setPageNumber(pageNumber)` → `{PCCViewer.ViewerControl}`

Sets the current page of the viewer to the specified page number.

Parameters:

Name	Type	Description
<code>pageNumber</code>	number string	The 1-based page number of the page. A string representation of a valid number is also accepted as a parameter.

Throws:

- If the parameter `pageNumber` is less than 1 or greater than the value returned by [PCCViewer.ViewerControl#getPageCount](#).

Type

Error

- If the parameter `pageNumber` is not a number or a string representation of a number.

Type

Error

- If the parameter `pageNumber` is not an integer page number.

Type

Error

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)

Example

```
// first create the pccViewer : Note: if the viewer
object has already been created, do not re-create it.
var viewerControl =
$("#viewer").pccViewer(viewerOptions).viewerControl;

var pageNumberSet = 2;
function pageChangedHandler(event) {
    var pageNumber = viewer.getPageNumber();
    if(pageNumber === pageNumberSet) {
        //now unsubscribe the event Note: do not
unsubscribe if future notifications are required.
        viewerControl.off("PageChanged",
pageChangedHandler);
        alert("Viewer was navigated to the desired page
successfully");
    }
}
// Subscribe to PageChanged event exposed by the API
viewerControl.on("PageChanged", pageChangedHandler);
viewerControl.setPageNumber(pageNumberSet);
```

setRedactionViewMode(redactionViewMode) → **{PCCViewer.ViewerControl}**

Sets the redaction view mode.

When set to "Draft" mode, this property will make visible the document content text underneath for the rectangle redaction and the text selection marks. In this mode, the redaction reasons will not be visible.

Setting the view mode to "Normal" will make the redaction rectangle marks and the text selection redactions opaque. In this mode, if the marks contain the redaction reasons then they will be visible.

Parameters:

Name	Type	Description
redactionViewMode	PCCViewer.RedactionViewMode	Displays redaction marks showing/hiding underneath document content text.

See: [PCCViewer.RedactionViewMode](#)

Throws:

If the `redactionViewMode` value is not one of the supported values. See [PCCViewer.ViewerControl~ViewerControlOptions](#) for mode details on the `redactionViewMode` initialization parameter.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)

Example

```
// show all the redaction rectangles and redaction text
selection marks showing underlying document text content.
viewerControl.setRedactionViewMode(PCCViewer.RedactionViewMo
```

`setScaleFactor(scaleFactor)` → `{PCCViewer.ViewerControl}`

Sets the scale factor of the viewer.

Note: The viewer has minimum and maximum scale limits. The limits depend on the size of the pages. check [PCCViewer.ViewerControl#getMinScaleFactor](#) and [PCCViewer.ViewerControl#getMaxScaleFactor](#) to determine the minimum and maximum scale.

Parameters:

Name	Type	Description
<code>scaleFactor</code>	number	A value indicating the scale factor to use in the viewer, with 1 being 100% zoom.

Throws:

- If the `scaleFactor` value is out of range.
- If the [PCCViewer.EventType.ViewerReady](#) event was not fired prior to calling this method.

Type

Error

Type

Error

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)

Example

```
viewerControl.setScaleFactor(.5); // Zoom the document to 50% of its actual size.
```

setSelectedConversation(conversation) → {PCCViewer.ViewerControl}

Sets the selected conversation.

Parameters:

Name	Type	Description
conversation	PCCViewer.Conversation	The conversation to select.

See: [PCCViewer.ViewerControl#getSelectedConversation](#)

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this property.

Type
Error

- If the `conversation` parameter is not an instance of [PCCViewer.Conversation](#).

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
// get the first selected Mark object
var theFirstMark = viewerControl.getSelectedMarks()[0];

// select the conversation of the Mark
viewerControl.setSelectedConversation(theFirstMark.getConver
```

`setSelectedSearchResult(searchResult, scrollToopt)` → `{PCCViewer.ViewerControl}`

Selects the specified search result and optionally navigates to the page of the search result.

If a value of `null` is passed in for the `searchResult` parameter, then any currently selected result will be cleared/deselected.

Parameters:

Name	Type	Attributes	Default	Description
<code>searchResult</code>	<code>PCCViewer.SearchResult</code> <code>null</code>			The search result object from the results object. If a value of <code>null</code> is passed, then no search results will be selected.
<code>scrollTo</code>	<code>boolean</code>	<optional>	<code>false</code>	If true, the viewer will navigate to the page with the search result. If the value of <code>searchResult</code> is <code>null</code> , then this argument is ignored.

See: [PCCViewer.ViewerControl#clearSearch](#)
[PCCViewer.ViewerControl#clearSelectedSearchResult](#)
[PCCViewer.ViewerControl#getSearchRequest](#)

Throws:

- If the `PCCViewer.EventType.ViewerReady` event was not fired prior to using this method.
 Type
 Error
- If the `searchResult` parameter is not an instance of `PCCViewer.SearchResult` or `null`.
 Type
 Error
- If the `searchResult` is not part of the currently known `PCCViewer.SearchRequest`, which is always the case if there is no current `PCCViewer.SearchRequest` object known to the viewer.
 Type
 Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
var searchRequest = viewerControl.search('Accusoft');

// add events to the search request
searchRequest.on("SearchCompleted", function(){
    // get the search results
    results = searchRequest.getResults();

    // set the result to the first
    viewerControl.setSelectedSearchResult(results[0],
true);
});
```

setSharpening(sharpeningFactor) → {PCCViewer.ViewerControl}

Sets the sharpening factor for the document. Note: This setting only affects viewing and is not persisted when printing or burning.

Parameters:

Name	Type	Description
sharpeningFactor	number	The sharpening factor to set the Sharpening for the document to.

Throws:

If the sharpening factor is not a valid integer between 0 and 100 inclusive.

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
`PCCViewer.ViewerControl`

Example

```
viewerControl.setSharpening(60);
```

setSvgLineWidthMultiplier(lineWidthMultiplier) → {PCCViewer.ViewerControl}

Numeric value which all SVG line width values should be multiplied by. Must be greater than 0 and must

not be greater than 100. To make lines twice as thick as they originally were, use a value of 2. To make lines half as thick as they originally were, use a value of 0.5, etc. To reset lines to their original thickness, use a value of 1. Note: This setting only affects viewing and is not persisted when printing or burning.

Parameters:

Name	Type	Description
lineWidthMultiplier	number	The line width multiplier for svg lines to set for the document.

Throws:

If amount is not greater than zero or is greater than 100

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
viewerControl.setSvgLineWidthMultiplier(50);
```

setViewMode(viewMode) → [{PCCViewer.ViewerControl}](#)

Sets the view mode.

When set to "Document" or "EqualFitPages", this property only has an effect on documents with different sized pages. Setting the view mode to "Document" maintains the relative size of each page when displaying a document. For example, if page 2 is smaller than page 1, it will appear smaller. Setting the view mode to "EqualFitPages" scales each page so that their width is the same. For example, if page 2 is smaller than page 1, it will be scaled larger so that its width is equal to the width of page 1.

Setting the view mode to "SinglePage" structures the viewer so that only a single page is shown at a time. Each page is scaled to fit within a view box, which is the initial size of the viewer and increases in size when zooming in (and decreases in size when zooming out). After the viewer initializes, the view mode may not be changed to or from SinglePage view mode (or an exception will occur).

Parameters:

Name	Type	Description
viewMode	PCCViewer.ViewMode	Display pages using this mode.

|

See: [PCCViewer.ViewMode](#)

Throws:

- If the `viewMode` value is unknown.
Type
Error
- If the [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.
Type
Error
- If setting the value to "SinglePage". This value must be set during initialization time. See [PCCViewer.ViewerControl~ViewerControlOptions](#) for mode details on the `viewMode` initialization parameter.
Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
viewerControl.setViewMode(PCCViewer.ViewMode.EqualFitPages);  
// Scale each page so that their width is the same.
```

`showMarks(marks)` → `{PCCViewer.ViewerControl}`

Shows the specified marks.

Parameters:

Name	Type	Description
marks	Array.< PCCViewer.Mark >	An Array of objects of type PCCViewer.Mark .

Throws:

- If [PCCViewer.EventType.ViewerReady](#) event was not fired prior to using this method.

Type
Error

- If any of the marks passed in are not valid objects of the type [PCCViewer.Mark](#).

Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type
[PCCViewer.ViewerControl](#)

Example

```
// shows all selected marks
viewerControl.showMarks(viewer.getSelectedMarks());
```

`validatePrintRange(range) → {boolean}`

Determines whether the range string is a valid page range to be used in [PCCViewer.ViewerControl#print](#).

Parameters:

Name	Type	Description
range	string	A string containing page numbers or ranges. See PCCViewer.ViewerControl#print .

Returns:

A value indicating whether the specified print range is valid.

Type
boolean

Example

```
viewerControl.validatePrintRange("1, 3-5"); // Returns
true if there are at least 5 pages in the document.
```

`validateSearch(searchQuery) → {Object}`

Validates each of the search terms in the `searchQuery` object. The validation process checks for valid custom regular expressions. This method is used by the default UI to filter out 'bad' pre-defined search terms.

Parameters:

Name	Type	Description
<code>searchQuery</code>	<code>string PCCViewer.ViewerControl~SearchQuery</code>	A string or a <code>SearchQuery</code> object that requires validation. See PCCViewer.ViewerControl#search .

Returns:

Returns an object with two summary properties and an array of objects that indicate whether each search term is valid. The array will be the same length as the array `searchQuery.searchTerms`. Objects contain the following properties:

- `errorsExist` {boolean} True if any validation errors exist. False if not.
- `summaryMsg` {string} (optional) A catch all message for cases where the search terms could not be reached for validation. This might occur if the `searchQuery` object is badly formed with improper key names or the viewer sends in an invalid object type (say boolean)
- array of objects:
 - `isValid` {boolean} Indicates if the search term of the matching index was valid.
 - `message` {string} Provides a human readable message indicating the reason the search term was invalid.

Type
Object

Example

```
var searchQuery = {
  searchTerms: [
    {
      searchTerm: '(?<=(category=))[a-z-]+', //
invalid regex
      contextPadding: 25,
      highlightColor: '#B22222',
      matchingOptions: {
        beginsWith: false,
        endsWith: false,
        exactPhrase: false,
        matchCase: true,
        matchWholeWord: false,
        wilddcard: false
      }
    }
  ]
}
```

```
}

viewerControl.validateSearch(searchQuery)

// returns
{
  errorsExist: true,
  searchTerms: [
    {
      isValid: false,
      message: "Search term must be a string
containing either plain text or a valid regular
expression."
    }
  ]
}
```

zoomIn(zoomFactor) → `{PCCViewer.ViewerControl}`

Zoom in on the document by the specified `zoomFactor`.

Note: The viewer has minimum and maximum scale limits. `zoomIn` will only change the viewer's scale up to, but not past, the maximum scale limit. `zoomIn` does not give an indication if the actual scale change was less than the requested zoom factor because the limit was reached. Instead, check [PCCViewer.ViewerControl#atMaxScale](#) to determine if the viewer is at max scale.

Parameters:

Name	Type	Description
<code>zoomFactor</code>	number	Zoom in by this factor. Valid values are between 1.01 and 20

See: [PCCViewer.ViewerControl#getAtMaxScale](#)
[PCCViewer.EventType.ScaleChanged](#)

Throws:

- If the `zoomFactor` value is invalid.
Type
Error
- If the [PCCViewer.EventType.ViewerReady](#) event was not fired prior to calling this method.
Type
Error

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)

Example

```
viewerControl.zoomIn(1.5); // Zoom in by 1.5x, or to
the maximum scale.
viewerControl.zoomIn(5); // Zoom in by 5x, or to the
maximum scale.

viewerControl.zoomIn(100); // Throws!
viewerControl.zoomIn(1); // Throws!
```

zoomOut(zoomFactor) → [{PCCViewer.ViewerControl}](#)

Zoom out on the document by the specified `zoomFactor`.

Note: The viewer has minimum and maximum scale limits. `zoomOut` will only change the viewer's scale down to, but not past, the minimum scale limit. `zoomOut` does not give an indication if the actual scale change was less than the requested zoom factor because the limit was reached. Instead, check [PCCViewer.ViewerControl#atMinScale](#) to determine if the viewer is at minimum scale.

Parameters:

Name	Type	Description
<code>zoomFactor</code>	number	Zoom out by this factor. Valid values are between 1.01 and 20.

See:

[PCCViewer.ViewerControl#getAtMinScale](#)
[PCCViewer.EventType.ScaleChanged](#)

Throws:

- If the `zoomFactor` value is invalid.
- If the [PCCViewer.EventType.ViewerReady](#) event has not fired prior to calling this method.

Type

Error

Type

Error

Returns:

The `ViewerControl` object on which this method was called.

Type

[PCCViewer.ViewerControl](#)**Example**

```
viewerControl.zoomOut(1.5); // Zoom out by 1.5x, or to
the minimum scale.
viewerControl.zoomOut(5); // Zoom out by 5x, or to
the minimum scale.

viewerControl.zoomOut(100); // Throws!
viewerControl.zoomOut(1); // Throws!
```

Type Definitions

ProximitySearchTerm

This class will hold all the necessary details to perform a proximity search. It will be used in the [PCCViewer.ViewerControl~SearchQuery](#) object that is passed to the [PCCViewer.ViewerControl#search](#) method.

Type:

- Object

Properties:

Name	Attributes	Description
<code>type</code> : string		This property is used by the API in order to determine that this search term is a proximity search term. The user will have to set the value to <code>proximity</code> in order for the term to be used by the proximity search algorithm.
<code>distance</code> : number		Number of intermediate words between the search terms.
<code>terms</code> : Array. < PCCViewer.ViewerControl~SearchTerm >		An array of search terms.
<code>contextPadding</code> : number	<optional> Default: 25	The maximum number of leading and trailing character in the context string (PCCViewer.SearchResult#getContext) that is given for a search result.
<code>highlightColor</code> : string	<optional>	The highlight color of the search results matching this search term (PCCViewer.SearchResult#getHighlightColor). If not defined, then a random color will be

Name	Attributes	Description
		chosen.

Example

```
// An example proximity search term object. Notice how
// you can specify any number of search terms with their own
// individual options.
var proximitySearchTerm = {
  type: "proximity",
  distance: 3,
  highlightColor: "#f73131",
  contextPadding: 30,
  terms: [{
    searchTerm: "influenza"
  }, {
    searchTerm: "infection",
    matchingOptions: {
      matchWholeWord: true
    }
  }, {
    searchTerm: "group",
    matchingOptions: {
      matchCase: false
    }
  }
]
};
// The proximity search term is used in a search query
// object.
searchRequest.on('SearchCompleted', function(){
  console.log();
});
// The proximity search term is used in a search query
// object.
// There can be many search term objects and proximity
// search term objects in a search query object.
var searchQuery = {
  searchTerms: [proximitySearchTerm]
};
// Execute the search
var searchRequest = viewerControl.search(searchQuery);
```

SaveMarkupOptions

The `PCCViewer.saveMarkup` API method takes a second optional parameter. This parameter is in the form of an object and provides options for saving marks to the server.

Type:

- Object

Properties:

Name	Attributes	Description
includeAnnotations : boolean	<optional> Default: true	Indicates whether annotation mark types should be saved.
includeSignatures : boolean	<optional> Default: false	Indicates whether signature mark types should be saved.
includeRedactions : boolean	<optional> Default: true	Indicates whether redaction mark types should be saved.

SearchMatchingOptions

This object is used to specify search options. It will be used in a [PCCViewer.ViewerControl~SearchTerm](#) object.

Type:

- Object

Properties:

Name	Attributes	Description
endsWith : boolean	<optional> Default: false	Match a phrase that ends with the search term. The matched phrase will be the shortest phrase that starts on a word boundary and ends with the matched phrase.
beginsWith : boolean	<optional> Default: false	Match a phrase that starts with the search term. The matched phrase will be the shortest phrase that starts with the matched phrase and ends on a word boundary. If this value is <code>true</code> , then the following options are ignored: <ul style="list-style-type: none"> • <code>endsWith</code>
exactPhrase : boolean	<optional> Default: true	Indicates whether the entire <code>searchTerm</code> is treated as a single phrase or if it is split on white space and individual words are matched based on the search options. If this value is <code>true</code> , then the following option is not supported: <ul style="list-style-type: none"> • <code>wildcard</code>
matchCase : boolean	<optional> Default:	Indicates whether matching is case sensitive.

Name	Attributes	Description
	false	
matchWholeWord : boolean	<optional> Default: false	Match a phrase that starts and ends on word boundaries. If this value is <code>true</code> , then the following options are ignored: <ul style="list-style-type: none"> • <code>wildcard</code> • <code>endsWith</code> • <code>beginsWith</code>
wildcard : boolean	<optional> Default: false	A value that indicates whether the search term includes wild cards. Supported wildcard characters, which may not be escaped, are: <ul style="list-style-type: none"> • '*' - match zero or more non-whitespace characters • '?' - match one character If this value is <code>true</code> , then the following options are ignored: <ul style="list-style-type: none"> • <code>endsWith</code> • <code>beginsWith</code>

Example

```
// A simple matching options object
var myMatchingOptions = {
    beginsWith: true,
};

// The matching options object is used in a search term
object
var searchTerm = {
    searchTerm: "sub",
    contextPadding: 25,
    highlightColor: '#B22222',
    matchingOptions: myMatchingOptions
};

// The search term object is used in the search query
object
var searchQuery = {
    searchTerms: [searchTerm]
};
```

SearchQuery

This object is used to specify one or more search terms and options for the [PCCViewer.ViewerControl#search](#) method.

Type:

- Object

Properties:

Name
searchTerms : Array. <(PCCViewer.ViewerControl~SearchTerm PCCViewer.ViewerControl~ProximitySearchTerm)>

Example

```
// An example search query object with one search term
var searchQuery = {
  searchTerms: [{
    searchTerm: "sub",
    contextPadding: 25,
    highlightColor: '#B22222',
    matchingOptions: {
      beginsWith: true,
    }
  }]
};
```

SearchTerm

This object is used to specify one search term and its options. It will be used in the [PCCViewer.ViewerControl~SearchQuery](#) object that is passed to the [PCCViewer.ViewerControl#search](#) method.

Type:

- Object

Properties:

Name	Attributes	Description
searchTerm : string		A string value to match against. treated as a string literal, a string or a regular expression. See search and PCCViewer.ViewerControl~Search
contextPadding : number	<optional> Default: 25	The maximum number of leading character in the context string (PCCViewer.SearchResult#getCo for a search result.

Name	Attributes	Description
<code>highlightColor : string</code>	<optional>	The highlight color of the search this search term (PCCViewer.SearchResult#getHig) If not defined, then a color will b different color will be chosen for and if <code>matchingOptions.exa</code> <code>false</code> , then a different color wil each word in the search term.
<code>matchingOptions :</code> PCCViewer.ViewerControl~SearchMatchingOptions	<optional>	Options that specify how the sea matched.
<code>searchTermIsRegex : boolean</code>	<optional> Default: <code>false</code>	Indicates if the search term is tre expression. If this value is <code>true</code> , then the fol options can be used: <ul style="list-style-type: none">• <code>matchingOptions.mat</code> Note that it is invalid to set the c options for a regular expression may make search behave in une:

Example

```
// An example search term object
var searchTerm = {
  searchTerm: "sub",
  contextPadding: 25,
  highlightColor: '#B22222',
  matchingOptions: {
    beginsWith: true,
  }
};

// The search term is used in a search query object.
// There can be many search term objects in a search
// query object.
var searchQuery = {
  searchTerms: [searchTerm]
};
```

ViewerControlOptions

These options are available and processed directly by the ViewerControl. When using the jQuery Plugin, [external:jQuery.fn#pccViewer](#), these options can be passed in along with [external:jQuery.fn~Options](#), and they will be passed into ViewerControl.

Type:

- Object

Properties:

Name	Attributes	Description
<code>documentID : string</code>		REQUIRED The ID of the document.
<code>imageHandlerUrl : string</code>		REQUIRED The end point of the services that support the viewer.
<code>language : Object</code>	<optional>	<p>Specifies the language to use for the ViewerControl.</p> <p>The options here are a subset of the external:jQuery.fn~Language options. These values are used directly by the ViewerControl, and are passed to the jQuery Plugin, external:jQuery.fn~Language. When creating a custom UI for the ViewerControl directly, these options need to be passed in during initialization. If the <code>language</code> element is not present in the following properties, ViewerControl will use the English language default.</p> <p>Properties</p> <ul style="list-style-type: none"> • <code>pageLoadFailed : string</code> Default: "Page Load Failed" Text to use for "Page Load Failed" • <code>pageLoadFailedRetry : string</code> <optional> Default: "Retry" Text to use for "Retry"
<code>viewMode : string</code>	<optional> Default: "Document"	The mode used to view documents of different sized pages. See the PCCViewer.ViewMode enumeration for details on each view mode.
<code>pageLayout : string</code>	<optional> Default: "Vertical"	The layout used to arrange the pages in the viewer. See the PCCViewer.ViewMode enumeration for details on each view mode.
<code>pageRotation : number</code>	<optional> Default: 0	The amount in degrees clockwise to rotate each page. Valid values are -270, -180, -90, 0, 90, 180.
<code>resourcePath : string</code>	<optional>	The location of the images used in the viewer.

Name	Attributes	Description
	Default: "img"	This is the folder that holds ArtMarkHandles.png and E files. This path should be re that the viewer is embedde
<code>printTemplate</code> : string	<optional> Default: ""	A text representation of a f for printing purposes. It is r this value be set to the con "printTemplate.html". If this empty string (default), then unavailable.
<code>template.print</code> : string	<optional> Default: ""	This is an alias for <code>printTe</code> <code>printTemplate</code> is not av <code>options</code> object, the contro property instead. This prop value as it is used in the jQuery.fn~Option
<code>encryption</code> : boolean	<optional> Default: false	Specifies whether the view encryption. See the help to Content Encryption" for mc
<code>serviceResponseTimeout</code> : number	<optional> Default: 60000	Indicates the response time services to the server. A val the default browser value v
<code>debug</code> : boolean	<optional> Default: false	Indicates whether the view actions.
<code>RedactionViewMode</code> : string	<optional> Default: "Normal"	The redaction view mode c document content text unc opaque rectangle redaction text selection redaction ma PCCViewer.RedactionView for details on redaction vie
<code>markHandleMode</code> : string	<optional> Default: "HideSideHandlesWhenClose"	The mark handle mode. See PCCViewer.MarkHandleMo details on each mark handl
<code>enableMultipleRedactionReasons</code> : boolean	<optional> Default: false	Specifies whether multiple are enabled. Set this prope order to use PCCViewer.Ma
<code>discardOutOfViewText</code> : boolean	<optional> Default: false	Specifies whether text on p displayed is discarded from that it is necessary to use tl PCCViewer.ViewerControl# method to request a text p any API that requires the te not currently displayed.
<code>searchMethodType</code> : string	<optional>	The type of search that will

Name	Attributes	Description
	Default: "auto"	<p>PCCViewer.ViewerControl#</p> <p>Note that client search will search is unavailable (this is using viewing packages). P</p> <ul style="list-style-type: none"> • "server" - Use the PCCViewer.ViewerCo API internally. • "client" - Use the PCCViewer.ViewerCo API internally. • "auto" - Viewer Con PCCViewer.ViewerCo when the page cour value set in the searchMethodPageC option. Viewer Cont PCCViewer.ViewerCo otherwise.
searchMethodPageCountThreshold : number	<optional> Default: 80	The maximum number of p can have for client-side sea used when searchMethodT Must be greater than 1.
viewerAssetsPath : number	<optional> Default: "viewer-assets"	The path to the viewer-asse used in the print template. {{viewerAssetsPath}} in you be replaced by the value sp option.

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:11 GMT-0400 (Eastern Daylight Time)

Mixin: Data

PCCViewer. Data

This object provides some helper methods that allow you to set and get data on any object.

Methods

getData(key) → {string|object}

Gets the data value for the given key, or gets a hash containing all key values, if a key was not provided.

Note: If a hash is returned, this will be a new object each time it is called. Adding new properties to the returned hash will not add data to the object.

Note: The returned data is not mutated or sanitized, which could lead to a security vulnerability if not sanitized properly before use.

Parameters:

Name	Type	Description
key	string	The key for which to get the data value.

See: [PCCViewer.Data#setData](#)
[PCCViewer.Data#getDataKeys](#)

Throws:

If the `key` argument is null or otherwise not a string.

Type
Error

Returns:

- If a key argument was provided, it returns the associated value.
- If a key argument was provided, but a value has not been associated with the key, then it returns undefined.
- If a key was not provided, it returns a hash object containing all key-value pairs.

Type
string | object

Example

```
// The key "Author" is set the value "Mark".
item.setData("Author", "Mark");

// The key "Note" is set the value "This is really
important!".
item.setData("Note", "This is really important!");

item.getData("Author"); // returns "Mark"
item.getData();         // returns {"Author":"Mark",
"Note":"This is really important!"}
item.getData("FooBar"); // returns undefined
```

`getDataKeys()` → {Array.<string>}

Gets an array of data keys known to this object.

See: [PCCViewer.Data#getData](#)
[PCCViewer.Data#setData](#)

Returns:

Returns an array of data keys known to this object. If no data is stored, then an empty array will be returned.

Type

Array.<string>

Example

```
// Returns an empty array before key-value pairs are
stored.
item.getDataKeys(); // returns []

// Returns a list of all keys.
item.setData("Author", "Mark");
item.setData("Note", "This is really important!");
item.getDataKeys(); // returns ["Author", "Note"]
```

setData(key, value) → {object}

Sets the data value for the given key.

Notes:

- Overwrites any data value already associated with the given key.
- There is no artificial limit imposed on the number of key-value pairs that are stored.
- If limits on the number of key-value pairs are required, they should be enforced by calling code.
- Setting the value as undefined results in no information for the key being persisted to the server.
- The returned data is not mutated or sanitized, which could lead to a security vulnerability if not sanitized properly before use.

Parameters:

Name	Type	Description
key	string	The key for which to set the data value.
value	string	This is the value to set for the key. <ul style="list-style-type: none">• This must be a string or undefined.• The maximum length of the string is not limited by this function.

See: [PCCViewer.Data#getData](#)

PCCViewer.Data#getDataKeys

Returns:

The object on which the method was called.

Type
object

Example

```
// Get data returns undefined before the key is set.
item.getData("Author"); // returns undefined

// The key "Author" is set the value "Mark".
item.setData("Author", "Mark");
item.getData("Author"); // returns "Mark"

// The key "Author" is overwritten with the value "Clark".
item.setData("Author", "Clark");
item.getData("Author"); // returns "Clark"

// The key "Author" is unset, by setting the value to
// undefined.
item.setData("Author", undefined);
item.getData("Author"); // returns undefined

// The value can only be set to a string or undefined.
// All other data types throw.
item.setData("FooBar", null); // throws
item.setData("FooBar", 1);    // throws
item.setData("FooBar", true); // throws
item.setData("FooBar", {});   // throws
item.setData("FooBar", []);   // throws
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Mixin: SessionData

PCCViewer. SessionData

This object provides some helper methods that allow you to set and get data on any object. In practice, any data saved using these methods will not be saved, it is purely for use during runtime.

Methods

`getSessionData(key) → {string|object}`

Gets the data value for the given key, or gets a hash containing all key values, if a key was not provided.

Note: While this is similar to [PCCViewer.Data#getData](#), the data stored in the session will not persist when the page is reloaded, it is used for runtime operations.

Note: If a hash is returned, this will be a new object each time it is called. Adding new properties to the returned hash will not add data to the object.

Note: The returned data is not mutated or sanitized, which could lead to a security vulnerability if not sanitized properly before use.

Parameters:

Name	Type	Description
key	string	The key for which to get the data value.

See: [PCCViewer.SessionData#setSessionData](#)
[PCCViewer.SessionData#getSessionDataKeys](#)

Throws:

If the `key` argument is null or otherwise not a string.

Type
Error

Returns:

- If a key argument was provided, it returns the associated value.
- If a key argument was provided, but a value has not been associated with the key, then it returns undefined.
- If a key was not provided, it returns a hash object containing all key-value pairs.

Type
string | object

Example

```
// The key "Author" is set the value "Mark".
item.setSessionData("Author", "Mark");

// The key "Note" is set the value "This is really
important!".
item.setSessionData("Note", "This is really important!");

item.getSessionData("Author"); // returns "Mark"
```

```
item.getSessionData(); // returns {"Author":"Mark",
"Note":"This is really important!"}
item.getSessionData("FooBar"); // returns undefined
```

`getSessionDataKeys()` → `{Array.<string>}`

Gets an array of session data keys known to this object.

Note: While this is similar to [PCCViewer.Data#getDataKeys](#), the data stored in the session will not persist when the page is reloaded, it is used for runtime operations.

See: [PCCViewer.SessionData#getSessionData](#)
[PCCViewer.SessionData#setSessionData](#)

Returns:

Returns an array of data keys known to this object. If no data is stored, then an empty array will be returned.

Type
Array.<string>

Example

```
// Returns an empty array before key-value pairs are
stored.
item.getSessionDataKeys(); // returns []

// Returns a list of all keys.
item.setSessionData("Author", "Mark");
item.setSessionData("Note", "This is really important!");
item.getSessionDataKeys(); // returns ["Author", "Note"]
```

`setSessionData(key, value)` → `{object}`

Sets the data value for the given key.

Note: While this is similar to [PCCViewer.Data#setData](#), the data stored in the session will not persist when the page is reloaded, it is used for runtime operations.

Notes:

- Overwrites any data value already associated with the given key.
- There is no artificial limit imposed on the number of key-value pairs that are stored.
- If limits on the number of key-value pairs are required, they should be enforced by calling code.
- Setting the value as undefined results in no information for the key being persisted to the server.
- The returned data is not mutated or sanitized, which could lead to a security vulnerability if not sanitized properly before use.

Parameters:

Name	Type	Description
key	string	The key for which to set the data value.
value	string	This is the value to set for the key. <ul style="list-style-type: none">• This must be a string or undefined.• The maximum length of the string is not limited by this function.

See: [PCCViewer.SessionData#getSessionData](#)
[PCCViewer.SessionData#getSessionDataKeys](#)

Returns:

The object on which the method was called.

Type
object

Example

```
// Get data returns undefined before the key is set.
item.getSessionData("Author"); // returns undefined

// The key "Author" is set the value "Mark".
item.setSessionData("Author", "Mark");
item.getSessionData("Author"); // returns "Mark"

// The key "Author" is overwritten with the value "Clark".
item.setSessionData("Author", "Clark");
item.getSessionData("Author"); // returns "Clark"

// The key "Author" is unset, by setting the value to
undefined.
item.setSessionData("Author", undefined);
item.getSessionData("Author"); // returns undefined

// The value can only be set to a string or undefined.
// All other data types throw.
item.setSessionData("FooBar", null); // throws
item.setSessionData("FooBar", 1); // throws
item.setSessionData("FooBar", true); // throws
item.setSessionData("FooBar", {}); // throws
item.setSessionData("FooBar", []); // throws
```


Namespace: Ajax

PCCViewer. Ajax

This object provides some helper methods that allow you to make and filter Ajax requests.

Members

headers :string

Gets or sets the headers that will be defined with every AJAX request Viewer Control makes.

This is an ECMA 5 property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Ajax#getHeaders](#)
[PCCViewer.Ajax#setHeaders](#)

Example

```
// get
var headers = PCCViewer.Ajax.headers;

// set
PCCViewer.Ajax.headers = { 'My-Secret-Header':
'mysecretkey' };
```

overrideMethod :string

Gets or sets the overrideMethod that will be defined with every AJAX request Viewer Control makes.

This is an ECMA 5 property that is defined only in browsers supporting ECMA 5. This property is not available in the older browsers like IE8. For the greatest browser compatibility, use the corresponding getter and setter methods.

Type:

- string

See: [PCCViewer.Ajax#getOverrideMethod](#)
[PCCViewer.Ajax#setOverrideMethod](#)

Example

```
// get
var overrideMethod = PCCViewer.Ajax.overrideMethod;

// set
PCCViewer.Ajax.overrideMethod = function(options) {
    options.headers['My-Secret-Header'] = 'mysecretkey';
};
```

Methods

`getHeaders()` → {Object}

Gets the headers object that will be sent with every request.

Note: Any header returned here will be sent with the request as long as the specific request doesn't use that same header with a different value. If you're expecting the header to exist on all requests, it should use a unique key that is not used by Accusoft. Certain mandatory headers required by every request will not be returned from this method. See [PCCViewer.Ajax#setHeaders](#) for this list of inaccessible headers.

See: [PCCViewer.Ajax#setHeaders](#)

Returns:

A copy of the headers object set with [PCCViewer.Ajax#setHeaders](#).

Type
Object

Example

```
// The default headers are set for every request
PCCViewer.Ajax.setHeaders({
    'My-Secret-Header': 'mysecretkey',
    'Authorization': 'Token asdf1234'
});

// Retrieve the headers that have previously been set
PCCViewer.Ajax.getHeaders(); // returns { 'My-Secret-Header': 'mysecretkey', 'Authorization': 'Token asdf1234' }
```

`getOverrideMethod()` → {function}

Gets the override method that will be called with every request.

Note: The function that is returned with this object is passed by reference and is the same one set with `setOverrideMethod`. Use caution when interacting with it.

See: [PCCViewer.Ajax#setOverrideMethod](#)

Returns:

The function defined with [PCCViewer.Ajax#setOverrideMethod](#).

Type
function

Example

```
// A function is set to add a header to every request
PCCViewer.Ajax.setOverrideMethod(function(options) {
    options.headers['My-Secret-Header'] = 'mysecretkey';
});

// Retrieve the override method that was previously set
PCCViewer.Ajax.getOverrideMethod(); // returns function
```

[setHeaders\(headers\)](#) → {Object}

Sets the headers object that will be sent with every request.

Notes:

- Overwrites any headers already set except the mandatory headers listed below.
- The mandatory headers are: Accusoft-Gid, Accusoft-Parent-Name, Accusoft-Parent-Pid, and Accusoft-Parent-Taskid.
- Setting the `headers` as undefined will result in no additional headers being sent with every request. The headers that the request requires will still be present, including the listed mandatory headers.

Parameters:

Name	Type	Description
headers	Object	An object containing the headers we will send with every request. <ul style="list-style-type: none">• This must be an object or undefined.

See: [PCCViewer.Ajax#getHeaders](#)

Throws:

- If `headers` is not an object or undefined.

Type
Error

- If `headers` attempts to overwrite a mandatory header. See the Notes section for more information.

Type
Error

Returns:

The `PCCViewer.Ajax` object.

Type
Object

Example

```
// getHeaders() returns an empty object if setHeaders() has
not been called
PCCViewer.Ajax.getHeaders(); // returns {}

// After calling setHeaders() with an object, getHeaders()
will return a copy of that object
PCCViewer.Ajax.setHeaders({ 'My-Secret-Header',
'mysecretkey' });
PCCViewer.Ajax.getHeaders(); // returns { 'My-Secret-
Header', 'mysecretkey' }

// After calling setHeaders() without any arguments,
getHeaders() will return an empty object
PCCViewer.Ajax.setHeaders();
PCCViewer.Ajax.getHeaders(); // returns {}

// The value can only be set to an object or undefined.
// All other data types throw.
PCCViewer.Ajax.setHeaders(null); // throws
PCCViewer.Ajax.setHeaders(1); // throws
PCCViewer.Ajax.setHeaders(true); // throws
PCCViewer.Ajax.setHeaders('b'); // throws
PCCViewer.Ajax.setHeaders([]); // throws
```

`setOverrideMethod(method) → {Object}`

Sets the override method that will be called with every request.

Notes:

- Overwrites any method already set.
- Setting the `method` as undefined will result in no override taking place. The request will proceed as usual

Parameters:

Name	Type	Description
method	PCCViewer.Ajax~OverrideMethod	A function that will be called with every request that allows you to override or modify the request. <ul style="list-style-type: none">• This must be a function or undefined.

See: [PCCViewer.Ajax#getOverrideMethod](#)
[PCCViewer.Ajax~OverrideMethod](#)

Throws:

If `method` is not a function or undefined.

Type
Error

Returns:

The [PCCViewer.Ajax](#) object.

Type
Object

Example

```
// getOverrideMethod() returns undefined if
setOverrideMethod() has not been called
PCCViewer.Ajax.getOverrideMethod(); // returns undefined

// After calling setOverrideMethod() with a function,
getOverrideMethod() will return the function by reference
PCCViewer.Ajax.setOverrideMethod(function(options) {
options.url = 'http://accusoft.com/'; });
PCCViewer.Ajax.getOverrideMethod(); // returns function

// After calling setOverrideMethod() without any arguments,
getOverrideMethod() will return undefined
PCCViewer.Ajax.setOverrideMethod();
PCCViewer.Ajax.getOverrideMethod(); // returns undefined

// The value can only be set to a function or undefined.
// All other data types throw.
PCCViewer.Ajax.setOverrideMethod(null); // throws
PCCViewer.Ajax.setOverrideMethod(1); // throws
PCCViewer.Ajax.setOverrideMethod(true); // throws
```

```
PCCViewer.Ajax.setOverrideMethod('b'); // throws
PCCViewer.Ajax.setOverrideMethod([]); // throws
```

Type Definitions

OverrideMethod(options) → {Promise|Undefined}

The function that you define with [PCCViewer.Ajax#setOverrideMethod](#) should follow this format

Parameters:

Name	Type	Description																																			
options	Object	<p>Properties</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Attributes</th> <th>Default</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>url</td> <td>String</td> <td></td> <td></td> <td></td> </tr> <tr> <td>method</td> <td>String</td> <td><optional></td> <td>"GET"</td> <td></td> </tr> <tr> <td>headers</td> <td>Object</td> <td><optional></td> <td>{}</td> <td></td> </tr> <tr> <td>body</td> <td>String</td> <td><optional></td> <td>null</td> <td></td> </tr> <tr> <td>timeout</td> <td>Number</td> <td><optional></td> <td>null</td> <td></td> </tr> <tr> <td>mimeType</td> <td>String</td> <td><optional></td> <td>null</td> <td> <ul style="list-style-type: none"> The mime type used for the response the server sends. See: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest </td> </tr> </tbody> </table>	Name	Type	Attributes	Default	Description	url	String				method	String	<optional>	"GET"		headers	Object	<optional>	{}		body	String	<optional>	null		timeout	Number	<optional>	null		mimeType	String	<optional>	null	<ul style="list-style-type: none"> The mime type used for the response the server sends. See: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest
Name	Type	Attributes	Default	Description																																	
url	String																																				
method	String	<optional>	"GET"																																		
headers	Object	<optional>	{}																																		
body	String	<optional>	null																																		
timeout	Number	<optional>	null																																		
mimeType	String	<optional>	null	<ul style="list-style-type: none"> The mime type used for the response the server sends. See: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest 																																	

See: [PCCViewer.Ajax#setOverrideMethod](#)

Returns:

- Optionally returns a then-able object (Promise) to prevent default execution from continuing.

Type

Promise | Undefined

Example

```
// This method can modify properties of the original
request
PCCViewer.Ajax.setOverrideMethod(function(options) {
    options.url = 'http://accusoft.com/';
});
```

```
// This method can prevent default execution from
continuing
PCCViewer.Ajax.setOverrideMethod(function(options) {
    var deferred = $.Deferred();

    $.ajax({
        url: options.url,
        method: options.method,
        headers: options.headers,
        data: options.body,
        mimeType: options.mimeType,
        timeout: options.timeout,
    }).then(
        function(data, textStatus, jqXHR) {
            deferred.resolve(new PCCViewer.AjaxResponse({
                status: jqXHR.status,
                textStatus: textStatus,
                headers: {
                    'Fake-Header': 'thisisnotreal'
                },
                responseText: jqXHR.responseText,
            }));
        },
        function(jqXHR, textStatus, errorThrown) {
            deferred.reject({
                error: new PCCViewer.Error('Error',
errorThrown),
                response: new PCCViewer.AjaxResponse({
                    status: jqXHR.status,
                    textStatus: textStatus,
                    headers: {
                        'Fake-Header': 'thisisnotreal'
                    },
                    responseText: jqXHR.responseText,
                })),
            });
        }
    );

    return deferred.promise();
});
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Namespace: Language

PCCViewer. Language

A global object that defines the language data used by the PrizmDoc Viewing Client.

Example

```
// The PCCViewer.Language object can be initialized from a
hash.
PCCViewer.Language.initializeData({
  "addComment": "Add Comment",
  "advancedSearch": "Advanced Search Options",
  "annotateLabel": "Annotate"
  // ...
});

// Later, we can get any data by key.
var myValue = PCCViewer.Language.getValue("annotateLabel");
```

Methods

getData() → {Object}

Returns the hash representing the language data.

Returns:

The hash representing the language data.

Type
Object

getValue(key) → {String|Object}

Gets the value from the Language data object with the given key or returns the key.

This method evaluates dots ('.') in the key, looking for a child object if a dot is seen. This method provides a convenience over directly accessing the data object because it will return the key instead of returning undefined or throwing in cases where an object is not defined.

Parameters:

Name	Type	Description
key	string	Look up language data for this key. The key is a string, which uses dot notation to specify sub-keys.

Throws:

If language data is undefined.

Type
Error

Returns:

Type
String | Object

Example

```
var myValue;  
  
// Look up language data by a string key  
myValue = PCCViewer.Language.getValue("annotateLabel");  
  
// Look up language data using a key with dot notation.  
myValue = PCCViewer.Language.getValue("printDialog.title");
```

initializeData(languageData)

Sets the language data from a hash.

Parameters:

Name	Type	Description
languageData	object	A hash representing the language data.

Example

```
// The PCCViewer.Language object can be initialized from a  
hash.  
PCCViewer.Language.initializeData({  
  "addComment": "Add Comment",  
  "advancedSearch": "Advanced Search Options",  
  "annotateLabel": "Annotate"  
  // ...  
});
```

Namespace: MouseTools

PCCViewer. MouseTools

The PCCViewer.MouseTools object allows you to create and get named mouse tools. This object encapsulates a collection of named mouse tools that are available to all viewer instances (globally). A mouse tool can be added to this collection using the [PCCViewer.MouseTools.createMouseTool](#) method, and a mouse tool in this collection can be accessed using the [PCCViewer.MouseTools.getMouseTool](#) method.

Methods

(static) [createMouseTool\(name, type\)](#) → {[PCCViewer.MouseTool](#)}

Create a new named mouse tool of a specific type.

- If the new name matches the name of an existing mouse tool of the same type, then all properties of the existing tool will be overwritten with the defaults. Subsequent calls to `.getMouseTool(name)` will return the tool with the new properties.
- If the new name already exists, but the type does not match, and error will be thrown.

Parameters:

Name	Type	Description
name	string	The name of the new mouse tool. This value is case-insensitive for comparison against existing mouse tools of the same name. Note: The case you provide will be persisted in the name property of the object that is returned, so it is best to pick a consistent naming scheme.
type	string	The type of the new mouse tool.

See: [PCCViewer.MouseTool.Type](#) for a list of possible mouse tool types.

Throws:

If calling this function using an existing `name` when the `type` does not match the already existing one.

Type
Error

Returns:

The MouseTool object that was created.

Type
[PCCViewer.MouseTool](#)

Example

```
// Create a new mouse tool with the name
"MyLineAnnotationMouseTool"
var myMouseTool =
PCCViewer.MouseTools.createMouseTool("MyLineAnnotationMouseTool",
"LineAnnotation");

// Configure the mouse tool or the template mark of the
mouse tool
myMouseTool.getTemplateMark().setOpacity(127);

// set the ViewerControl to use the mouse tool
viewerControl.setCurrentMouseTool("MyLineAnnotationMouseTool");
```

(static) `getMouseTool(name)` → {`PCCViewer.MouseTool`|undefined}

Gets a named mouse tool.

Parameters:

Name	Type	Description
name	string	The name of the <code>MouseTool</code> to get. This value is case-insensitive.

Returns:

The `MouseTool` object with the specified name, or `undefined`, if a `MouseTool` with the specified name does not exist.

Type

`PCCViewer.MouseTool` | `undefined`

Example

```
var mouseToolName = "FooMouseTool";

// get the mouse tool
var mouseTool =
PCCViewer.MouseTools.getMouseTool(mouseToolName);

// check that the named mouse tool actually exists
if (mouseTool) {
    // do something with the MouseTool, you can do
different things based on the type of the tool
    switch (mouseTool.getType()) {
```

```
        case PCCViewer.MouseTool.Type.LineAnnotation:
            ...
            break;
        default:
            ...
    }
}
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Namespace: Signatures

PCCViewer. Signatures

An instance of [PCCViewer.ObservableCollection](#). This object is a common management utility used to keep track of the currently known signatures.

See: [PCCViewer.ObservableCollection](#) for methods and events available on this collection.

Members

(inner) [FreehandSignature](#)

A plain object convention describing a freehand drawn signature.

Properties:

Name	Attributes	Description
<code>type</code> : string		Describes the type of data in this signature. For FreehandSignature , this value will always be <code>path</code> .
<code>path</code> : string		The path data of the signature.
<code>width</code> : number		The absolute width of the path data.
<code>height</code> : number		The absolute height of the path data.
<code>category</code> : string undefined	<optional>	The category of this signature. See signatureCategories in external:jQuery.fn~Options for more information.

(inner) [TextSignature](#)

A plain object convention describing a text based signature.

Properties:

Name	Attributes	Description
<code>type</code> : string		Describes the type of data in this signature. For <code>TextSignature</code> , this value will always be <code>text</code> .
<code>text</code> : string		The text contents of the signature.
<code>category</code> : string undefined	<optional>	The category of this signature. See <code>signatureCategories</code> in external:jQuery.fn~Options for more information.

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:11 GMT-0400 (Eastern Daylight Time)

Namespace: Util

PCCViewer. Util

This object provides some common helper functions.

Methods

(static) `calculateNonOverlappingSelections(selections, backgroundColor)` → `{Array.<selection>}`

This method takes several selection objects -- ranges of start indices and lengths -- and determines if and split overlapping ranges into separate selections objects. It will also interpret the new color in the overlapping regions using [PCCViewer.Util.layerColors](#). The non-overlapping selections from this function should be used when highlighting text inside marks to ensure the best display. See [PCCViewer.Mark#highlightText](#).

Parameters:

Name	Type	Description
selections	Array. <Object> Object	<p>An array of objects or a single object that defines a highlight.</p> <p>Each object has the following properties:</p> <ul style="list-style-type: none"> • <code>startIndex</code> {number} - required <ul style="list-style-type: none"> ◦ The start index of the selection, in a 0-based index of string characters. ◦ The valid range is <code>startIndex >= 0</code>. • <code>length</code> {number} - required <ul style="list-style-type: none"> ◦ The length of the selection, in characters. ◦ The valid range is <code>length > 0</code>. • <code>color</code> {string} - required <ul style="list-style-type: none"> ◦ Specifies the Hexadecimal color for the selection. ◦ Valid values are any 7-character string representing a color. The first letter must be a "#" symbol and the other six characters must be hexadecimal digits representing the red, green, and blue portions of the color. • <code>opacity</code> {number} - required <ul style="list-style-type: none"> ◦ Specifies the opacity of the selection. ◦ Valid values are from 0 to 255 (inclusive).
backgroundColor	string	A valid 7-character Hexadecimal color string. The first letter must be a "#" symbol and the other six characters must be hexadecimal digits representing the red, green, and blue portions of the color.

See: [PCCViewer.Mark#highlightText](#)
[PCCViewer.Util.layerColors](#)

Throws:

- If any of the selection objects have an invalid `startIndex` number, or the value is undefined.

Type
Error

- If any of the selection objects have an invalid `length` number, or the value is undefined.

Type
Error

- If any of the selection objects have an invalid `color` Hex string, or the value is undefined.

Type
Error

- If any of the selection objects have an invalid `opacity` number, or the value is undefined.

Type
Error

- If the backgroundColor argument is not a valid Hex string, or the value is undefined.

Type
Error

Returns:

A collection of non-overlapping selection objects, representing the same selections that were passed into the function.

Type
Array.<selection>

(static) `convertPageRangeToArray(range, optionsopt)` → {Array.<number>}

Converts the supplied page range to an array, where each element in the array is a page number. The returned array will be sorted ascending by page number and will not contain duplicates.

Parameters:

Name	Type	Attributes	Description															
range	string		<p>A string specifying page numbers or ranges.</p> <p>Valid values are any string using this format (specified using EBNF):</p> <pre> range = "all" pageRange; pageRange = pageNumber, [",", range] subRange, [",", range]; pageNumber = naturalNumber; subRange = pageNumber, "-", pageNumber; </pre> <p>For example: "all", "1,2,3", and "1, 5-10" are all valid ranges.</p>															
options	object	<optional>	<p>An object specifying validation options.</p> <p>Properties</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Attributes</th> <th>Default</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>lowerLimit</td> <td>number</td> <td><optional></td> <td>1</td> <td>the lower limit (inclusive) of the valid range.</td> </tr> <tr> <td>upperLimit</td> <td>number</td> <td><optional></td> <td>Number.MAX_VALUE</td> <td>the upper</td> </tr> </tbody> </table>	Name	Type	Attributes	Default	Description	lowerLimit	number	<optional>	1	the lower limit (inclusive) of the valid range.	upperLimit	number	<optional>	Number.MAX_VALUE	the upper
Name	Type	Attributes	Default	Description														
lowerLimit	number	<optional>	1	the lower limit (inclusive) of the valid range.														
upperLimit	number	<optional>	Number.MAX_VALUE	the upper														

Name	Type	Attributes	Description
			limit (inclusive) of the valid range.
		allowEmpty	boolean <optional> false Indicates that an empty range string is valid.

See: [PCCViewer.Util.validatePageRange](#)

Throws:

- If pageRange does not conform to the supported format.
Type
Error
- If pageRange specifies a range that is not within the bounds of options.lowerLimit and options.upperLimit.
Type
Error

Returns:

An array containing an element for each page number specified in the range.

Type
Array.<number>

Example

```
PCCViewer.Util.convertPageRangeToArray("1, 3-5"); //
returns [1, 3, 4, 5]

PCCViewer.Util.convertPageRangeToArray("1, 3-5", {
  lowerLimit: 1,
  upperLimit: 6
}); // returns [1, 3, 4, 5]

PCCViewer.Util.convertPageRangeToArray("1, 3-100", {
```



```

    lowerLimit: 1,
    upperLimit: 6
}); // throws because the range goes beyond the upper limit

```

(static) layerColors(colors, backgroundColor) → {string}

This method takes an ordered list of colored layers, and calculates the flat color resulting from stacking all the layers.

Note: this stacking order calculation uses the W3C specification for simple alpha compositing, and is therefore not a complete color mixing solution. Instead, it calculates RGB color composition to browser specification.

Parameters:

Name	Type	Description
colors	Array. <Object> Object	<p>An array of objects or a single object that defines a color layer. Note that the first element in the array will be the topmost layer in the stacking order, and the last element will be the bottom-most layer.</p> <p>Each object has the following properties:</p> <ul style="list-style-type: none"> • color {string} - required <ul style="list-style-type: none"> ◦ Specifies the Hexadecimal color for the highlight. ◦ Valid values are any 7-character string representing a color. The first letter must be a "#" symbol and the other six characters must be hexadecimal digits representing the red, green, and blue portions of the color. • opacity {number} - required <ul style="list-style-type: none"> ◦ Specifies the opacity of the highlight. ◦ Valid values are from 0 to 255 (inclusive).
backgroundColor	string	A valid 7-character Hexadecimal color string. The first letter must be a "#" symbol and the other six characters must be hexadecimal digits representing the red, green, and blue portions of the color.

Throws:

- If any of the color objects have an invalid `color` Hex string, or the value is undefined.

Type
Error

- If any of the color objects have an invalid `opacity` number, or the value is undefined.

Type
Error

- If the `backgroundColor` argument is not a valid Hex string, or the value is undefined.

Type
Error

Returns:

The resulting flat color produced by stacking all of the layers on top of the background color. This value will be a 7-character Hexadecimal color.

Type
string

(static) save(filename, stringValue)

Triggers file saving functionality with data generated on the client side. This method handles browser-specific differences in file generation. As such, it may have slightly different behavior in the various browsers. The end result provides a common interface for triggering a file save.

Parameters:

Name	Type	Description
filename	string	The desired name of the output file. This will be used as the name or suggested name in browsers that support it.
stringValue	string	The data, in string format, to be written to the file.

(static) validatePageRange(range, options_{opt}) → {boolean}

Determines whether the range string is a valid page range of the format supported by the viewer. It will optionally validate that the range is within a specified set of limits.

Parameters:

Name	Type	Attributes	Description
range	string		<p>A string specifying page numbers or ranges.</p> <p>Valid values are any string using this format (specified using EBNF):</p> <pre> range = "all" pageRange; pageRange = pageNumber, [",", range] subRange, [",", range]; pageNumber = naturalNumber; subRange = pageNumber, "-", pageNumber; </pre> <p>For example: "all", "1,2,3", and "1, 5-10" are all valid ranges.</p>
options	object	<optional>	<p>An object specifying validation options.</p> <p>Properties</p>

Name	Type	Attributes	Description				
			Name	Type	Attributes	Default	Description
			lowerLimit	number	<optional>	1	the lower limit (inclusive) of the valid range.
			upperLimit	number	<optional>	Number.MAX_VALUE	the upper limit (inclusive) of the valid range.
			allowEmpty	boolean	<optional>	false	Indicates that an empty range string is valid.

See: [PCCViewer.Util.convertPageRangeToArray](#)

Returns:

A value indicating whether the specified page range is valid.

Type
boolean

Example

```
PCCViewer.Util.validatePageRange("1, 3-5", {
    lowerLimit: 1,
    upperLimit: 6
}); // returns true

PCCViewer.Util.validatePageRange("1, 3-5, 100", {
    lowerLimit: 1,
    upperLimit: 6
}); // returns false

PCCViewer.Util.validatePageRange("this is not a valid
range", {
    lowerLimit: 1,
    upperLimit: 6
}); // returns false
```

```
PCCViewer.Util.validatePageRange("3", {
  lowerLimit: 1,
  upperLimit: myViewerControl.getPageCount() // assuming
myViewerControl is a PCCViewer.ViewerControl
}); // returns true if the document has at least 3 pages

PCCViewer.Util.validatePageRange(""); // returns false

PCCViewer.Util.validatePageRange("", {
  allowEmpty: true
}); // returns true
```

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:11 GMT-0400 (Eastern Daylight Time)

Namespace: MarkSchema

MarkSchema

The following data structures will be used for serializing and deserializing marks into a JSON format. This documentation represents the Object Schema for [all marks](#). To ease these, properties that are present in each mark are documented under the name [Mark](#), and properties related to each individual mark are documented using that mark's type.

Some other generic types relevant to all, or multiple, marks are documented as [PageData](#), [Comment](#), [Conversation](#), [Rectangle](#), [Point](#), and [LineGroup](#).

Type Definitions

EllipseAnnotation

The [EllipseAnnotation](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
rectangle : MarkSchema~Rectangle	The position of the mark on the page.
pageData : MarkSchema~PageData	The size of the page that the mark is on.
borderColor : String	A 6-character hexadecimal color string, including the # sign.
borderThickness : Number	The thickness of the border in pixels.

Name	Description
<code>fillColor</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>opacity</code> : Number	The opacity of the mark, from 0 to 255.

Implements: [MarkSchema~Mark](#)

FreehandAnnotation

The [FreehandAnnotation](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>rectangle</code> : MarkSchema~Rectangle	The position of the mark on the page.
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.
<code>color</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>opacity</code> : Number	The opacity of the mark, from 0 to 255.
<code>path</code> : String	An SVG-style path, using M, L, and C commands.
<code>thickness</code> : Number	The thickness of the border in pixels.

Implements: [MarkSchema~Mark](#)

FreehandSignature

The [FreehandSignature](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>rectangle</code> : MarkSchema~Rectangle	The position of the mark on the page.
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.
<code>color</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>horizontalAlignment</code> : String	A value from PCCViewer.Mark.HorizontalAlignment .
<code>path</code> : String	An SVG-style path, using M, L, and C commands.

Name	Description
<code>thickness</code> : Number	The thickness of the border in pixels.

Implements: [MarkSchema~Mark](#)

HighlightAnnotation

The [HighlightAnnotation](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>fillColor</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>startIndex</code> : Number	The character index of the start of the selection.
<code>selectedText</code> : String	The selected text.
<code>textLength</code> : Number	The length of the selected text.
<code>lineGroups</code> : Array. < MarkSchema~LineGroup >	The individual line rectangles that make up the selection.

Implements: [MarkSchema~Mark](#)

ImageStampAnnotation

The [ImageStampAnnotation](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>rectangle</code> : MarkSchema~Rectangle	The position of the mark on the page.
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.
<code>imageDataUrl</code> : String.<base64>	The base64 encoded image data.
<code>imageId</code> : String	The ID associated with the image.

Implements: [MarkSchema~Mark](#)

ImageStampRedaction

The [ImageStampRedaction](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>rectangle</code> : MarkSchema~Rectangle	The position of the mark on the page.
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.
<code>imageDataUrl</code> : <code>String.<base64></code>	The base64 encoded image data.
<code>imageId</code> : <code>String</code>	The ID associated with the image.

Implements: [MarkSchema~Mark](#)

LineAnnotation

The [LineAnnotation](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.
<code>endPoint</code> : MarkSchema~Point	The end of the line.
<code>startPoint</code> : MarkSchema~Point	The start of the line.
<code>color</code> : <code>String</code>	A 6-character hexadecimal color string, including the # sign.
<code>endHeadType</code> : <code>String</code>	A value from PCCViewer.Mark.LineHeadType .
<code>opacity</code> : <code>Number</code>	The opacity of the mark, from 0 to 255.
<code>thickness</code> : <code>Number</code>	The thickness of the line in pixels.

Implements: [MarkSchema~Mark](#)

PolylineAnnotation

The [PolylineAnnotation](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.
<code>points</code> : Array.< MarkSchema~Point >	The array of points making up the line.
<code>color</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>opacity</code> : Number	The opacity of the mark, from 0 to 255.
<code>thickness</code> : Number	The thickness of the line in pixels.

Implements: [MarkSchema~Mark](#)

RectangleAnnotation

The [RectangleAnnotation](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>rectangle</code> : MarkSchema~Rectangle	The position of the mark on the page.
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.
<code>borderColor</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>borderThickness</code> : Number	The thickness of the border in pixels.
<code>fillColor</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>opacity</code> : Number	The opacity of the mark, from 0 to 255.

Implements: [MarkSchema~Mark](#)

RectangleRedaction

The [RectangleRedaction](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>rectangle</code> : MarkSchema~Rectangle	The position of the mark on the page.
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.
<code>borderColor</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>borderThickness</code> : Number	The thickness of the border in pixels.
<code>fillColor</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>fontColor</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>reason</code> : String	The redaction reason for this redaction mark.
<code>reasons</code> : Array.<String>	The redaction reasons for this redaction mark.

Implements: [MarkSchema~Mark](#)

StampAnnotation

The [StampAnnotation](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>rectangle</code> : MarkSchema~Rectangle	The position of the mark on the page.
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.
<code>color</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>label</code> : String	The text to display inside the stamp.

Implements: [MarkSchema~Mark](#)

StampRedaction

The [StampRedaction](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>rectangle</code> : MarkSchema~Rectangle	The position of the mark on the page.

Name	Description
pageData : MarkSchema~PageData	The size of the page that the mark is on.
label : String	The text to display inside the stamp.

Implements: [MarkSchema~Mark](#)

StrikethroughAnnotation

The [StrikethroughAnnotation](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
color : String	A 6-character hexadecimal color string, including the # sign.
thickness : Number	The thickness of the line in pixels.
textLength : Number	The length of the selected text.
startIndex : Number	The character index of the start of the selection.
selectedText : String	The selected text.
lineGroups : Array. < MarkSchema~LineGroup >	The individual line rectangles that make up the selection.

Implements: [MarkSchema~Mark](#)

TextAnnotation

The [TextAnnotation](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
rectangle : MarkSchema~Rectangle	The position of the mark on the page.
pageData : MarkSchema~PageData	The size of the page that the mark is on.
borderColor : String	A 6-character hexadecimal color string, including the # sign.
borderThickness : Number	The thickness of the border in pixels.

Name	Description
<code>fillColor</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>fontColor</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>fontName</code> : String	The name of the font to use for the mark.
<code>fontSize</code> : Number	The size of the font, in pixels.
<code>fontStyle</code> : Array.<String>	An array of values any from PCCViewer.Mark.FontStyles .
<code>maxLength</code> : Number	The maximum number of characters allowed.
<code>horizontalAlignment</code> : String	A value from PCCViewer.Mark.HorizontalAlignment .
<code>text</code> : String	The text of the mark.
<code>opacity</code> : Number	The opacity of the mark, from 0 to 255.

Implements: [MarkSchema~Mark](#)

TextAreaSignature

The [TextAreaSignature](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>rectangle</code> : MarkSchema~Rectangle	The position of the mark on the page.
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.
<code>fontColor</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>fontName</code> : String	The name of the font to use for the mark.
<code>fontStyle</code> : Array.<String>	An array of values any from PCCViewer.Mark.FontStyles .
<code>horizontalAlignment</code> : String	A value from PCCViewer.Mark.HorizontalAlignment .
<code>maxFontSize</code> : Number	The maximum size of the font, in pixels.
<code>maxLength</code> : Number	The maximum number of characters allowed.
<code>text</code> : String	The text of the mark.

Implements: [MarkSchema~Mark](#)

TextHyperlinkAnnotation

The [TextHyperlinkAnnotation](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>fillColor</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>textLength</code> : Number	The length of the selected text.
<code>startIndex</code> : Number	The character index of the start of the selection.
<code>selectedText</code> : String	The selected text.
<code>lineGroups</code> : Array. < MarkSchema~LineGroup >	The individual line rectangles that make up the selection.
<code>href</code> : String	The URL that the link points to.

Implements: [MarkSchema~Mark](#)

TextInputSignature

The [TextInputSignature](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>rectangle</code> : MarkSchema~Rectangle	The position of the mark on the page.
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.
<code>mask</code> : MarkSchema~Mask	An input mask that will be displayed in the mark to assist the user from inputting undesirable characters.
<code>fontColor</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>fontName</code> : String	The name of the font to use for the mark.
<code>maxLength</code> : Number	The maximum number of characters allowed.
<code>text</code> : String	The text of the mark.
<code>horizontalAlignment</code> : String	A value from PCCViewer.Mark.HorizontalAlignment .

Implements: [MarkSchema~Mark](#)

TextRedaction

The [TextRedaction](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>rectangle</code> : MarkSchema~Rectangle	The position of the mark on the page.
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.
<code>fontColor</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>fontName</code> : String	The name of the font to use for the mark.
<code>fontSize</code> : Number	The size of the font, in pixels.
<code>maxLength</code> : Number	The maximum number of characters allowed.
<code>horizontalAlignment</code> : String	A value from PCCViewer.Mark.HorizontalAlignment .
<code>text</code> : String	The text of the mark.

Implements: [MarkSchema~Mark](#)

TextSelectionRedaction

The [TextSelectionRedaction](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>lineGroups</code> : Array. < MarkSchema~LineGroup >	The individual line rectangles that make up the selection.
<code>reason</code> : String	The redaction reason for this redaction mark.
<code>reasons</code> : Array.<String>	The redaction reasons for this redaction mark.
<code>selectedText</code> : String	The selected text.
<code>startIndex</code> : Number	The character index of the start of the selection.
<code>textLength</code> : Number	The length of the selected text.

Implements: [MarkSchema~Mark](#)

TextSignature

The [TextSignature](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>rectangle</code> : MarkSchema~Rectangle	The position of the mark on the page.
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.
<code>color</code> : String	A 6-character hexadecimal color string, including the # sign.
<code>fontName</code> : String	The name of the font to use for the mark.
<code>horizontalAlignment</code> : String	A value from PCCViewer.Mark.HorizontalAlignment .
<code>text</code> : String	The text of the mark.

Implements: [MarkSchema~Mark](#)

TransparentRectangleRedaction

The [TransparentRectangleRedaction](#) object.

Type:

- [MarkSchema~Mark](#)

Properties:

Name	Description
<code>rectangle</code> : MarkSchema~Rectangle	The position of the mark on the page.
<code>pageData</code> : MarkSchema~PageData	The size of the page that the mark is on.

Implements: [MarkSchema~Mark](#)

Comment

A mark comment.

Type:

- Object

Properties:

Name	Description
data : Object.<key, string>	A property bag of user-defined values.
creationDateTime : String	An ISO string of the created time.
text : String	The text of the comment.

Conversation

A collection of comments related to a mark.

Type:

- Object

Properties:

Name	Description
comments : Array.<MarkSchema~Comment>	The comments associated with the conversation.
data : Object.<key, string>	A property bag of user-defined values.

LineGroup

A collection of line groups.

Type:

- Object

Properties:

Name	Description
pageNumber : Number	The page number of the line group.
pageData : MarkSchema~PageData	The page size of the page where that particular line group appears.
startIndex : Number	The character index of the start of this group.
length : Number	The length of characters in this group.
lines : Array.<MarkSchema~Rectangle>	One or more rectangles that appear on the page, as part of the selection.

Mark

All marks will have the following properties.

Properties:

Name	Description
<code>uid</code> : String	A global unique ID for this mark, to identify it across the system.
<code>type</code> : String	A value from PCCViewer.Mark.Type denoting the mark type.
<code>pageNumber</code> : Number	The page that the mark is located on.
<code>creationDateTime</code> : String	An ISO string of the created time.
<code>modificationDateTime</code> : String	An ISO string of the last modified time.
<code>interactionMode</code> : String	A value from PCCViewer.Mark.InteractionMode .
<code>data</code> : Object.<string, string>	A property bag of user-defined values.
<code>conversation</code> : MarkSchema~Conversation	The conversation object.

Mask

An object defining the mask for this mark.

Type:

- Object

Properties:

Name	Description
<code>value</code> : String	The string representation of the mask. The user input will <i>look</i> like this string once they have finished their input. Each character in this string that does not have a translation will be represented to the user literally.
<code>translations</code> : Object	The translations to use for the given mask value. The key represents a character present in the mask value, and the value is a regular expression which validates the acceptable user input for that character.

PageData

An object providing metadata about the page at the time that the mark was saved.

Type:

- Object

Properties:

Name	Description
<code>width</code> : Number	The width of the page at the time the mark was saved.
<code>height</code> : Number	The height of the page at the time the mark was saved.

Point

A point, defining the coordinates from the top-left of the page in pixels.

Type:

- Object

Properties:

Name	Description
<code>x</code> : Number	The distance from the left edge of the page.
<code>y</code> : Number	The distance from the top edge of the page.

Rectangle

A rectangle or bounding rectangle, defining the top-left corner relative to the top-left of the page, as well as the width and height, in pixels.

Type:

- Object

Properties:

Name	Description
<code>x</code> : Number	The left side of the rectangle, relative to the left page edge.
<code>y</code> : Number	The top of the rectangle, relative to the top page edge.
<code>width</code> : Number	The width of the bounding rectangle.
<code>height</code> : Number	The height of the bounding rectangle.

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

Namespace: MarkupLayerSchema

MarkupLayerSchema

The following data structures will be used for serializing and deserializing markup layers into a JSON format. This documentation represents the Object Schema for markup layers.

Properties:

Name	Description
<code>name</code> : string	The name of the markup layer.
<code>originalXmlName</code> : string	The name of the web tier XML record from which the marks of this layer were originally stored.
<code>data</code> : string	A property bag of user-defined values.
<code>marks</code> : Array	An array of MarkSchema~Mark objects. Note that comments on marks in this layer are stored in this comments array and not stored under the particular mark.
<code>comments</code> : Array	An array of MarkupLayerSchema~Comment objects for marks in this layer and in other layers. Note that comments on marks in this layer are stored in this comments array and not stored under the particular mark in the marks array.

Type Definitions

Comment

A mark comment.

Type:

- Object

Properties:

Name	Description
<code>markUid</code> : string	The global unique ID for the mark the comment is under.
<code>data</code> : Object.<key, string>	A property bag of user-defined values.
<code>creationDateTime</code> : String	An ISO string of the created time.
<code>text</code> : String	The text of the comment.

Documentation generated by [JSDoc 3.3.3](#) on Thu Apr 22 2021 12:32:10 GMT-0400 (Eastern Daylight Time)

E-Signature Controls

Overview

This section contains the following information:

- External: jQuery.fn
- Class: ESigner
- Class: TemplateDesigner
- Module: button-set
- Module: checkbox-collection
- Module: data-persist
- Module: date-picker
- Module: download-signed-form-trigger
- Module: download-signed-form
- Module: dropdown
- Module: event-store
- Module: field-edit
- Module: field-list
- Module: fill-checklist
- Module: fill-form-controller
- Module: fill-main-toolbar
- Module: fill-progress
- Module: form-controller
- Module: form-extraction
- Module: form-summary
- Module: form-tools
- Module: global-settings-menu
- Module: global-settings-trigger
- Module: keyboard-controller
- Module: multiple-selection
- Module: notification
- Module: page-navigation
- Module: profile-manager
- Module: state-store
- Module: svg-icons
- Module: template-io
- Module: template-manager
- Module: template-name-header
- Module: text-input
- Module: zoom-fit

External: jQuery.fn

External: jQuery.fn

jQuery.fn

The jQuery plugin namespace.

See: [jQuery Plugins](#)

Members

(inner) `configParameters`

Properties:

Name	Type	Attributes	Default	Description
documentID	string			The viewingSessionId to use in ViewerControl. This value is always required.
templateDocumentId	string	<optional>		This parameter may be used when creating a template in order to associate the template for use with a document. When opening a form, the value of this parameter will be used to identify the document associated with the formDefinitionId. This value is required by the TemplateDesigner.
formDefinitionId	string	<optional>		The formDefinitionId to be used when opening a form. Specifying this value in the TemplateDesigner will cause it to open the specific form definition, while leaving the value out will cause it to create a new form. This value is required by the ESigner.
formRoleId	string	<optional>		The formRoleId to be used when

Name	Type	Attributes	Default	Description
				opening a form. Specifying this v in the ESigner w cause it to only create fields ass to the given forr role.
signatureDateFormat	external:"jQuery.fn"~DateFormat	<optional>	"MM/DD/YYYY"	Specifies the da format that is st when saving a template. When e-signer loads a template, it disp date signatures the date format stored in the template.
language	object			Specifies the language to use the text in the vi Use this option localize the view This property sh be set to the contents of the "viewer-assets/language US.json". Both th signer and the template-design ship with their o version of the language file. Ea viewer has a diff set of language strings in the language file. This value is always required.
imageHandlerUrl	string	<optional>	"../pcc.ashx"	The end point o web tier service support the view
onViewerCreation	function	<optional>		This function is available for bot ESigner and the TemplateDesign will trigger when

Name	Type	Attributes	Default	Description
				viewer's DOM is ready to use. You use one parameter inside this function will be an ESigner or a TemplateDesigner object depending on which viewing class you are using.

(inner) [DateFormat](#) :String

The format to use when displaying a date. The table below outlines the supported date format tokens and provides example output.

	Token	Output
Month	M	1 2 ... 11 12
	MM	01 02 ... 11 12
Day	D	1 2 ... 30 31
	DD	01 02 ... 30 31
Year	YY	70 71 ... 29 30
	YYYY	1970 1971 ... 2029 2030
Hour	H	0 1 ... 22 23
	HH	00 01 ... 22 23
	h	1 2 ... 11 12
	hh	01 02 ... 11 12
Minute	m	0 1 ... 58 59
	mm	00 01 ... 58 59
AM/PM	A	AM PM
	a	am pm

Type:

- String

Methods

[pccESigner\(options\)](#)

A jQuery plugin to initialize a new [ESigner](#) viewer.

Parameters:

Name	Type	Description
options	external:"jQuery.fn"~configParameters	The configuration options used to initialize the viewer.

See: [ESigner](#) for an example.

pccTemplateDesigner(options)

A jQuery plugin to initialize a new [TemplateDesigner](#) viewer.

Parameters:

Name	Type	Description
options	external:"jQuery.fn"~configParameters	The configuration options used to initialize the viewer.

See: [TemplateDesigner](#) for an example.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Class: ESigner

Class: ESigner

(protected) **ESigner()**

(protected) **new ESigner()**

The e-signer viewer constructor.

Requires:

- [module:event-store](#)
- [module:state-store](#)
- [module:page-navigation](#)
- [module:zoom-fit](#)
- [module:template-io](#)
- [module:fill-form-controller](#)
- [module:template-name-header](#)
- [module:svg-icons](#)
- [module:data-persist](#)

- [module:profile-manager](#)
- [module:fill-main-toolbar](#)
- [module:fill-checklist](#)
- [module:fill-progress](#)
- [module:download-signed-form-trigger](#)
- [module:download-signed-form](#)
- [module:date-picker](#)
- [module:keyboard-controller](#)
- [module:notification](#)
- [module:text-input](#)
- [module:dropdown](#)
- [module:button-set](#)
- [module:checkbox-collection](#)

Example

```
var viewer = $('#pcc-viewer').pccESigner(options);
```

Requires

- [module:event-store](#)
- [module:state-store](#)
- [module:page-navigation](#)
- [module:zoom-fit](#)
- [module:template-io](#)
- [module:fill-form-controller](#)
- [module:template-name-header](#)
- [module:svg-icons](#)
- [module:data-persist](#)
- [module:profile-manager](#)
- [module:fill-main-toolbar](#)
- [module:fill-checklist](#)
- [module:fill-progress](#)
- [module:download-signed-form-trigger](#)
- [module:download-signed-form](#)
- [module:date-picker](#)
- [module:keyboard-controller](#)
- [module:notification](#)
- [module:text-input](#)
- [module:dropdown](#)
- [module:button-set](#)
- [module:checkbox-collection](#)

Members

checklist : [module:fill-checklist](#)

Type:

- [module:fill-checklist](#)

dataPersist : [module:data-persist](#)

Type:

- [module:data-persist](#)

`datePicker` :[module:date-picker](#)

Type:

- [module:date-picker](#)

`downloadSignedForm` :[module:download-signed-form](#)

Type:

- [module:download-signed-form](#)

`downloadSignedFormTrigger` :[module:download-signed-form-trigger](#)

Type:

- [module:download-signed-form-trigger](#)

`eventStore` :[module:event-store](#)

Type:

- [module:event-store](#)

`fillProgress` :[module:fill-progress](#)

Type:

- [module:fill-progress](#)

`formController` :[module:form-controller](#)

Type:

- [module:form-controller](#)

`formSummary` :[module:form-summary](#)

Type:

- [module:form-summary](#)

`keyboardController` :`module:keyboard-controller`

Type:

- `module:keyboard-controller`

`mainToolBar` :`module:fill-main-toolbar`

Type:

- `module:fill-main-toolbar`

`notification` :`module:notification`

Type:

- `module:notification`

`pageNavigation` :`module:page-navigation`

Type:

- `module:page-navigation`

`profileManager` :`module:profile-manager`

Type:

- `module:profile-manager`

`stateStore` :`module:state-store`

Type:

- `module:state-store`

`templateIO` :`module:template-io`

Type:

- `module:template-io`

`templateNameHeader` :`module:template-name-header`

Type:

- [module:template-name-header](#)

`zoomFit` : [module:zoom-fit](#)

Type:

- [module:zoom-fit](#)

Methods

`destroy()`

Destroys the viewer and all modules, and returns the parent DOM element to its original state.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Class: TemplateDesigner

Class: TemplateDesigner

(protected) `TemplateDesigner()`

(protected) `new TemplateDesigner()`

The template designer viewer constructor.

Requires:

- [module:event-store](#)
- [module:state-store](#)
- [module:form-tools](#)
- [module:page-navigation](#)
- [module:zoom-fit](#)
- [module:form-controller](#)
- [module:field-list](#)
- [module:field-edit](#)
- [module:multiple-selection](#)
- [module:notification](#)
- [module:template-io](#)
- [module:template-manager](#)
- [module:svg-icons](#)

- [module:text-input](#)
- [module:dropdown](#)
- [module:button-set](#)
- [module:checkbox-collection](#)
- [module:keyboard-controller](#)

Example

```
var viewer = $('#pcc-viewer').pccTemplateDesigner(options);
```

Requires

- [module:event-store](#)
- [module:state-store](#)
- [module:form-tools](#)
- [module:page-navigation](#)
- [module:zoom-fit](#)
- [module:form-controller](#)
- [module:field-list](#)
- [module:field-edit](#)
- [module:multiple-selection](#)
- [module:notification](#)
- [module:template-io](#)
- [module:template-manager](#)
- [module:svg-icons](#)
- [module:text-input](#)
- [module:dropdown](#)
- [module:button-set](#)
- [module:checkbox-collection](#)
- [module:keyboard-controller](#)

Members

eventStore :[module:event-store](#)

Type:

- [module:event-store](#)

fieldEdit :[module:field-edit](#)

Type:

- [module:field-edit](#)

fieldList :[module:field-list](#)

Type:

- [module:field-list](#)

`formController` :`module:form-controller`

Type:

- `module:form-controller`

`formExtraction` :`module:form-extraction`

Type:

- `module:form-extraction`

`formTools` :`module:form-tools`

Type:

- `module:form-tools`

`globalSettingsMenu` :`module:global-settings-menu`

Type:

- `module:global-settings-menu`

`globalSettingsTrigger` :`module:global-settings-trigger`

Type:

- `module:global-settings-trigger`

`keyboardController` :`module:keyboard-controller`

Type:

- `module:keyboard-controller`

`multipleSelection` :`module:multiple-selection`

Type:

- `module:multiple-selection`

`notification` :`module:notification`

Type:

- [module:notification](#)

`pageNavigation` :[module:page-navigation](#)

Type:

- [module:page-navigation](#)

`stateStore` :[module:state-store](#)

Type:

- [module:state-store](#)

`templateIO` :[module:template-io](#)

Type:

- [module:template-io](#)

`templateManager` :[module:template-manager](#)

Type:

- [module:template-manager](#)

`zoomFit` :[module:zoom-fit](#)

Type:

- [module:zoom-fit](#)

Methods

`destroy()`

Destroys the viewer and all modules, and returns the parent DOM element to its original state.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: button-set

Module: button-set

A set of UI buttons that interact as one group. The Button Set supports two modes: toggling the buttons between the on and off state, as well as toggling through an arbitrary list of ordered values.

In the on/off mode, each button will be turned to on when clicked. When one button is selected, it turns all other buttons that are in the on state to off.

In the arbitrary toggle mode, each button will cycle through its own toggle values. When one button is selected, it will remove the active toggle value from any other button that currently has one. Those buttons will be reset, and begin the cycle at the first toggle value the next time they are clicked.

Examples

```
<!-- In this example, the buttons toggle on and off -->

<!--The following HTML includes a couple button set
components with the same data-pcc-name
so that they are included in the same set. An element is
specified as a button set by setting
the data-pcc-component attribute to "buttonset".-->
<button data-pcc-component="buttonset"
        data-pcc-name="mousetools"
        data-pcc-value="SignatureTemplate"
        class="pcc-button">
  <span data-pcc-icon="pcc-icon-signature"></span>
  <label>Signature</label>
</button>
<button data-pcc-component="buttonset"
        data-pcc-name="mousetools"
        data-pcc-value="InitialsTemplate"
        class="pcc-button">
  <span data-pcc-icon="pcc-icon-initials"></span>
  <label>Initials</label>
</button>
```

```
<!-- In this example, the buttons toggle among arbitrary
values -->
<button data-pcc-component="buttonset"
        data-pcc-name="mousetools"
        data-pcc-value="SignatureTemplate"
        data-pcc-toggle="on,sticky"
        class="pcc-button">
  <span data-pcc-icon="pcc-icon-signature"></span>
  <label>Signature</label>
</button>
```

```
// Require the button set module.
var ButtonSet = require('../elements/button-set.js');
```

```
var mySet;

// Pass each button set element to the button set module to
// initialize each button.
// parent is the element that contains the button set
// elements.
$(parent).find('[data-pcc-
component="buttonset"]').each(function() {
    // ButtonSet will return the entire set of buttons that
    // have been added
    // using the same 'data-pcc-name' value
    mySet = ButtonSet(this);
});

mySet.on('change', function(ev, data) {
    // data about the buttonset
    console.log(data);
});
```

(require("button-set"))(e1) → {HTMLElement}

Parses and initializes a button set.

Parameters:

Name	Type	Description
e1	HTMLElement	The parent element in which to parse for the button set component.

Returns:

The parsed button set element.

Type
HTMLElement

Members

(static) pccElements :Object

The button elements in the button set.

Type:

- Object

off :module:event-store~off

Removes an event handler from the button set.

Type:

- module:event-store~off

on :module:event-store~on

Registers an event handler on the button set.

Type:

- module:event-store~on

Methods

destroy()

Destroys the button set component.

value(val) → {Object}

Gets or sets the value of the button set. The values are specified in the HTML for each button using the data-pcc-value attribute.

Parameters:

Name	Type	Description
val	string	The value of the button to make active.

Returns:

The button set element if a value is passed. Otherwise, the current value is returned.

Type
Object

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: checkbox-collection

Module: checkbox-collection

A set of UI checkboxes that interact as one group.

Examples

```
<!--The following HTML includes a checkbox collection
component containing a single checkbox.
An element is specified as a checkbox collection by setting
the data-pcc-component attribute
to "checkboxcollection".-->
<span data-pcc-component="checkboxcollection"
      data-pcc-name="required"
      data-pcc-value="required"
      data-pcc-label="Required"
      class="pcc-margin"></span>
```

```
// Require the checkbox collection module.
var CheckboxCollection = require('../elements/checkbox-
collection.js');

// Pass each checkbox collection element to the checkbox
collection module to initialize each checkbox.
// parent is the element that contains the checkbox
collection element.
$(parent).find('[data-pcc-
component="checkboxcollection"]').each(function() {
    CheckboxCollection(this);
});
```

(require("checkbox-collection"))(el) → {HTMLElement}

Parses and initializes a checkbox collection.

Parameters:

Name	Type	Description
el	HTMLElement	The parent element in which to parse for the checkbox collection component.

Returns:

The parsed checkbox collection element.

Type

HTMLElement

Members

`(static) pccElements :Object`

The checkbox elements in the checkbox collection.

Type:

- Object

`off :module:event-store~off`

Removes an event handler from the checkbox collection.

Type:

- module:event-store~off

`on :module:event-store~on`

Registers an event handler on the checkbox collection.

Type:

- module:event-store~on

Methods

`destroy()`

Destroys the checkbox collection component.

`value(val) → {Object}`

Gets or sets the values of the checkbox collection. The values are specified in the HTML for each checkbox using the data-pcc-value attribute.

Parameters:

Name	Type	Description
val	Array	An array of values to check the corresponding checkbox elements.

Returns:

The checkbox collection element if a value is passed. Otherwise, an array of the values that correspond to the currently checked checkboxes is returned.

Type

Object

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: data-persist

Module: data-persist

Provides the ability to store state data in the browser's local storage.

Note: this module is an example of a persistence module. It presents potential security concerns, in that it may allow users to store sensitive information in non-secure browser storage. Please make sure this module fits your security model before using it in production.

```
(require("data-persist"))(viewer)
```

Created the data persistence module.

Parameters:

Name	Type	Description
viewer	Core	The core viewer to which the module will attach.

Listens to Events:

- [module:event-store#event:PersistSignatures](#)

Example

```
var DataPersist = require('data-persist.js');  
  
// a generic Viewer constructor  
var myDataPersist = DataPersist(viewer);
```

Methods

[destroy\(\)](#)

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: date-picker

Module: date-picker

Provides a date picker UI.

Listens to Events:

- [module:event-store#event:CreateDate](#)
- [module:event-store#event:FormatDate](#)
- [module:event-store#event:StateModified](#) for "FocusField" state.

Example

```
var DatePicker = require('date-picker.js');

// a generic Viewer constructor
var myDatePicker = DatePicker(this, {
  dateFormat: 'MM/DD/YYYY'
});
```

`(require("date-picker"))(viewer, options)`

Created the template name header module.

Parameters:

Name	Type	Description									
viewer	Core	The core viewer to which the module will attach.									
options	Object	An options object. <table border="1" data-bbox="400 1610 1399 1756"> <thead> <tr> <th colspan="3">Properties</th> </tr> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>dateFormat</td> <td>external:"jQuery.fn"~DateFormat</td> <td>The format string to use when providing the selected date.</td> </tr> </tbody> </table>	Properties			Name	Type	Description	dateFormat	external:"jQuery.fn"~DateFormat	The format string to use when providing the selected date.
Properties											
Name	Type	Description									
dateFormat	external:"jQuery.fn"~DateFormat	The format string to use when providing the selected date.									

Methods

`destroy()`

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: download-signed-form-trigger

Module: download-signed-form-trigger

Triggers the event for downloading a signed form.

This UI of this module is a button that the user can click to start burning the fields into the form and then download the burned document. This button will be disabled until the user has filled all of the required fields on the document.

Fires:

- `module:event-store#event:StartBurningForm` - This event is fired to indicate that the user wants to begin burning the document and download the burned document.

Listens to Events:

- `module:event-store#event:StateModified` for "FieldList" state.

Example

```
var DownloadSignedFormTrigger = require('download-signed-form-trigger.js');

// a generic Viewer constructor
function Viewer(opts) {
    var myDownloadSignedFormTrigger =
    DownloadSignedFormTrigger(this, {
        elem:
    document.getElementById('myDownloadSignedFormTrigger')
    });
}
```

`(require("download-signed-form-trigger"))(viewer, options)`

Creates the download signed form trigger UI module.

Parameters:

Name	Type	Description
<code>viewer</code>	Core	The core viewer to which the module will attach.
<code>options</code>	Object	An options object.

Name	Type	Description
Properties		
Name	Type	Description
elem	HTMLElement	The element in which the module UI will be inserted.

Methods

destroy()

Destroys the module.

Documentation generated by *JSDoc 3.5.5* on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: download-signed-form

Module: download-signed-form

Manages downloading a signed form.

This UI of this module is a modal dialog box that shows the signature burn-in status, allows the user to cancel the burning and download, and allows the user to download the signed form.

`(require("download-signed-form"))(viewer, options)`

Creates the download signed form module.

Parameters:

Name	Type	Description									
viewer	Core	The core viewer to which the module will attach.									
options	Object	An options object. <table border="1"> <thead> <tr> <th colspan="3">Properties</th> </tr> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>elem</td> <td>HTMLElement</td> <td>The element in which the module UI will be inserted.</td> </tr> </tbody> </table>	Properties			Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Properties											
Name	Type	Description									
elem	HTMLElement	The element in which the module UI will be inserted.									

Listens to Events:

- `module:event-store#event:BurnForm` - The download signed form dialog will be displayed when

this event is triggered.

- [module:event-store#event:DisplayForm](#) - Gets the form name from this event.

Example

```
var DownloadSignedForm = require('download-signed-form.js');

// a generic Viewer constructor
function Viewer(opts) {
  var myDownloadSignedForm = DownloadSignedForm(this, {
    elem:
document.getElementById('myDownloadSignedForm')
  });
}
```

Methods

[destroy\(\)](#)

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: dropdown

Module: dropdown

A dropdown menu.

Examples

```
<!--The following HTML includes a dropdown component. An
element is specified
as a dropdown by setting the data-pcc-component attribute
to "dropdown".
The dropdown options must be included as children elements,
where the
data-pcc-value attribute is used to specify the value of
each option.-->
<div class="pcc-select" data-pcc-component="dropdown" data-
pcc-name="font" data-pcc-default="Arial">
  <div data-pcc-value="Arial">Arial</div>
```



```
<div data-pcc-value="Comic Sans">Comic Sans</div>
<div data-pcc-value="Courier">Courier</div>
<div data-pcc-value="Courier New">Courier New</div>
<div data-pcc-value="Geneva">Geneva</div>
<div data-pcc-value="Georgia">Georgia</div>
<div data-pcc-value="Helvetica">Helvetica</div>
<div data-pcc-value="Times">Times</div>
<div data-pcc-value="Times New Roman">Times New
Roman</div>
  <div data-pcc-value="Verdana">Verdana</div>
</div>
```

```
// Require the dropdown module.
var Dropdown = require('../elements/dropdown.js');

// Pass each dropdown element to the dropdown module to
initialize each dropdown.
// parent is the element that contains the dropdown
element.
$(parent).find('[data-pcc-
component="dropdown"]').each(function() {
  Dropdown(this);
});
```

(require("dropdown"))(e1) → {HTMLElement}

Parses and initializes a dropdown component.

Parameters:

Name	Type	Description
e1	HTMLElement	The parent element in which to parse for the dropdown component.

Returns:

The parsed dropdown element.

Type
HTMLElement

Members

off :module:event-store~off

Removes an event handler from the dropdown.

Type:

- module:event-store~off

on :module:event-store~on

Registers an event handler on the dropdown.

Type:

- module:event-store~on

Methods

addOption(val, style_{opt}) → {Object}

Adds an option to the list of values in the dropdown.

Parameters:

Name	Type	Attributes	Description
val	string		The value of the dropdown option.
style	string	<optional>	The style of the dropdown option.

Returns:

The dropdown element.

Type
Object

destroy()

Destroys the dropdown component.

value(val) → {Object}

Gets or sets the value of the dropdown. The values are specified in the HTML for each dropdown option using the data-pcc-value attribute.

Parameters:

Name	Type	Description
val	string	The value of the dropdown option to select.

Returns:

The dropdown element if a value is passed. Otherwise, the currently selected value is returned.

Type
Object

`valueList()` → {Array}

Gets an array of the values of the dropdown options.

Returns:

An array of the values of the dropdown options.

Type
Array

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: event-store

Module: event-store

An event API. This event store is used internally by the viewer, and should not need to be initialized outside of that usage.

Example

```
var EventStore = require('event-store.js');

// a generic Viewer constructor
function Viewer(opts) {
  // other modules will expect this to be present
  this.eventStore = EventStore(this);
}
```

Events

AccessGlobalSettings

Indicates that the user needs to access the global settings dialog for modifying the various global settings available for the templates.

See: `module:event-store~eventCallback`

AlignFields

Triggers alignment of the given fields.

Properties:

Name	Type	Description
alignment	string	The plane and direction of the alignment operation.
markIds	Array	An array of Mark IDs to align.

See: `module:event-store~eventCallback`

Example

```
viewer.eventStore.trigger('AlignFields', {  
  alignment: 'horizontal-left',  
  markIds: viewer.viewerControl.getSelectedMarks()  
});
```

BurnForm

Indicates that the user wants to burn the signatures to the form.

See: `module:event-store~eventCallback`

CreateDate

Triggers a user request to select a date.

Properties:

Name	Type	Attributes	Description															
position	Object		The position to locate the UI, relative to a rectangle on the window. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>x</td><td>number</td><td>The x-axis location.</td></tr><tr><td>y</td><td>number</td><td>The y-axis location.</td></tr><tr><td>width</td><td>number</td><td>The width of the rectangle.</td></tr><tr><td>height</td><td>number</td><td>The height of the rectangle.</td></tr></tbody></table>	Name	Type	Description	x	number	The x-axis location.	y	number	The y-axis location.	width	number	The width of the rectangle.	height	number	The height of the rectangle.
Name	Type	Description																
x	number	The x-axis location.																
y	number	The y-axis location.																
width	number	The width of the rectangle.																
height	number	The height of the rectangle.																

Name	Type	Attributes	Description
onDone	string	<optional>	An event name to trigger, in the style of <code>module:event-store~onDoneCallback</code> . It should provide a date string as the event's <code>data</code> attribute.

See: `module:event-store~eventCallback`
`module:event-store~onDoneCallback`

CreateSignature

Triggers a user request to apply a signature to a field.

Properties:

Name	Type	Attributes	Description
category	string	<optional>	The field type that this signature belongs to. If undefined, no category will be assigned. Known values for this are <code>signature</code> and <code>initials</code> .
signatureType	string	<optional>	The type of signature being created, as represented by the target mark. Possible values are <code>FreehandSignature</code> and <code>TextSignature</code> . When this value is not defined, a good experience would be to allow the user to choose.
onDone	string	<optional>	An event name to trigger, in the style of <code>module:event-store~onDoneCallback</code> . It should provide a signature object as the <code>data</code> attribute with the data returned from <code>PCCViewer.SignatureControl</code> , or undefined to signal that the user cancelled the action.

See: `module:event-store~eventCallback`
`module:event-store~onDoneCallback`

DeleteFields

Triggers deletion of the given fields.

Properties:

Name	Type	Description
markIds	Array	An array of Mark IDs to delete.

See: `module:event-store~eventCallback`

Example

```
viewer.eventStore.trigger('DeleteFields', {
  markIds: [1, 2, 3]
});
```

DeselectAllTemplateFields

Indicates that all previously selected fields are now deselected.

See: [module:event-store~eventCallback](#)

DisplayForm

Indicates that a [module:state-store~FormDefinition](#) is available for displaying on the document.

Properties:

Name	Type	Description
formDefinition	module:state-store~FormDefinition	Provides the form definition as the data parameter.

See: [module:event-store~eventCallback](#)

Example

```
viewer.eventStore.on('DisplayForm', function(ev,
formDefinition) {
  // logic to convert the form data to visible marks on
the document
  createMarksForFormData(formDefinition.formData);
});
```

DuplicateFields

Triggers duplication of the given fields.

Properties:

Name	Type	Description
markIds	Array	An array of Mark IDs to duplicate.

See: [module:event-store~eventCallback](#)

Example

```
viewer.eventStore.trigger('DuplicateFields', {
  markIds: [1, 2, 3]
});
```

FocusChecklistItem

Indicates that an item in the checklist has been focused.

Properties:

Name	Type	Attributes	Description
markId	string	<optional>	The ID of the mark that corresponds to the focused checklist item.

See: [module:event-store~eventCallback](#)

FormatDate

Triggers a request to format a given date using the default format.

Properties:

Name	Type	Attributes	Description
data	Date		The Date object to convert to the signatureDateFormat. This format can be defined in the form definition. If signatureDateFormat is not defined, MM/DD/YYYY is used.
onDone	string	<optional>	An event name to trigger, in the style of module:event-store~onDoneCallback. It should provide a date string as the event's data attribute.

See: [module:state-store~FieldList](#)
[external:"jQuery.fn"~DateFormat](#)
[module:event-store~eventCallback](#)
[module:event-store~onDoneCallback](#)

FormCopied

Indicates that the current form has successfully been copied.

See: [module:event-store~eventCallback](#)

FormLoaded

Indicates that a form has been loaded from the server.

See: [module:event-store~eventCallback](#)

KeyCombinationsTriggered

Indicates that the user pressed the keyboard key combinations.

Properties:

Name	Type	Description						
state	string	This should always be defined as the string "KeyCombinationsTriggered".						
stateValue	Object	An object containing data about the key combination. Properties <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>keyCombinations</td> <td>string</td> <td>A string property containing keyboard keyCombinations that were pressed by the user.</td> </tr> </tbody> </table>	Name	Type	Description	keyCombinations	string	A string property containing keyboard keyCombinations that were pressed by the user.
Name	Type	Description						
keyCombinations	string	A string property containing keyboard keyCombinations that were pressed by the user.						

See: [module:event-store~eventCallback](#)
[module:event-store#event:RegisterKeyCombinations](#)

ManageSignatures

Triggers a user request to manage signatures in a list.

Properties:

Name	Type	Attributes	Description
category	string	<optional>	The field type that this signature belongs to. When defined, only the signatures belonging to that category should be displayed. If undefined, all known signatures should be displayed. Known values for this are <code>signature</code> and <code>initials</code> .
selectedSignature	Object	<optional>	An object, consisting of the signature data (as returned by <code>PCCViewer.SignatureControl</code>), defining the signature that the user has selected. Unless the user changes the signature, this data should be returned in the <code>onDone</code> event as the signature data.
onDone	string	<optional>	An event name to trigger, in the style of <code>module:event-store~onDoneCallback</code> . It should provide a signature object as the <code>data</code> attribute with the data returned from <code>PCCViewer.SignatureControl</code> , or undefined to signal that the user has removed the selected signature.

See: [module:event-store~eventCallback](#)

module:event-store~onDoneCallback

MatchSizeFields

Triggers changing width or height depending on direction (horizontal/vertical) of all selected fields to match that dimension of a field, selected first.

Properties:

Name	Type	Description
markIds	Array	An array of Mark IDs to apply the change to.
direction	string	Determines which dimension (width or height) to change.

See: module:event-store~eventCallback

Example

```
viewer.eventStore.trigger('MatchSizeFields', {
  markIds: [1, 2, 3],
  direction: 'vertical'
});
```

ModifyMultipleTemplateFields

Indicates that multiple template fields need to be modified.

Properties:

Name	Type	Description
markIds	array	An array of currently selected Mark IDs.

See: module:event-store~eventCallback

ModifyState

Indicates that a module or external code would like to modify a known state value. Generally, a module should not listen to this event. It is handled by StateStore, which will in turn fire [module:event-store#event:StateModified](#), to notify all other modules that a new state value is available.

Properties:

Name	Type	Attributes	Default	Description
state	string			The name of the state being modified.
stateValue	*			The new value of the state.

Name	Type	Attributes	Default	Description
operation	string	<optional>	"extend"	Specifies how the modification should occur. By default, any modification will extend the current state, merging any additional values from <code>stateValue</code> into the current state that is stored in the State Store. You can also specify "replace" as the operation value, causing the old state to be discarded, and the exact value of <code>stateValue</code> to become the current state.

See: [module:state-store](#)
[module:event-store#event:StateModified](#)
[module:event-store~eventCallback](#)

ModifyTemplateField

Indicates that a template field needs to be modified.

Properties:

Name	Type	Description
markId	string	The ID of the mark that corresponds to the template field.

See: [module:event-store~eventCallback](#)

Notify

Triggers a notification.

Properties:

Name	Type	Attributes	Default	Description
type	String			The type of notification, either "error" or "success".
message	String			The message of the notification.
displayTime	Number	<optional>	0	The amount of time (in milliseconds) to display the notification. If a positive number is not specified, then the notification is displayed until the user closes it.

See: [module:event-store~eventCallback](#)

Example

```
viewer.eventStore.trigger('Notify', {
  type: 'error',
  message: 'An error occurred.'
})
```

```
});
```

PersistSignatures

Triggers a manual save of the signatures. It will save all signatures currently in the `PCCViewer.Signatures` collection.

See: [module:event-store~eventCallback](#)

RegisterKeyCombinations

Requests registration of a new keyboard key combination.

Properties:

Name	Type	Description						
state	string	This should always be defined as the string "KeyCombinations".						
stateValue	Object	An object containing data about the key combination. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>keyCombinations</td><td>string</td><td>Keyboard key combinations when pressed would trigger KeyCombinationsTriggered event.</td></tr></tbody></table>	Name	Type	Description	keyCombinations	string	Keyboard key combinations when pressed would trigger KeyCombinationsTriggered event.
Name	Type	Description						
keyCombinations	string	Keyboard key combinations when pressed would trigger KeyCombinationsTriggered event.						

See: [module:event-store~eventCallback](#)
[module:event-store#event:KeyCombinationsTriggered](#)

SaveTemplate

Indicates that the user needs to save the form template in its current state.

See: [module:event-store~eventCallback](#)

SaveTemplateCopy

Indicates that the user needs to save a new copy of the form template in its current state. This event is implemented to convert the viewer into using the newly created copy when the copying is complete.

See: [module:event-store~eventCallback](#)

SelectTemplateField

Indicates that a template field needs to be selected.

Properties:

Name	Type	Description
markId	string	The ID of the mark that corresponds to the template field.

See: [module:event-store~eventCallback](#)

StateModified

Indicates that a state value has been modified. This event should only be fired by the StateStore module. It has the following data properties:

Properties:

Name	Type	Description
state	string	The name of the state that was modified.
stateValue	*	The current value of the state that was modified.

See: [module:state-store](#)
[module:event-store#event:ModifyState](#)
[module:event-store~eventCallback](#)

Example

```
viewer.eventStore.on('StateModified', function(ev, data){
  if (data.state !== 'MyStateKey') { return; }

  // handle the state change here
});
```

TemplateSaved

Indicates that a template successfully saved.

See: [module:event-store~eventCallback](#)

Example

```
viewer.eventStore.trigger('TemplateSaved');
```

TemplateSaveFailed

Indicates that a template failed to save.

See: [module:event-store~eventCallback](#)

Example

```
viewer.eventStore.trigger('TemplateSaveFailed');
```

ToggleChecklist

Triggers the checklist to toggle open or closed.

See: [module:event-store~eventCallback](#)

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: field-edit

Module: field-edit

Provides UI showing the settings of a form field and allowing the user to edit the form field.

Fires:

- [module:event-store#event:ModifyState](#)
- [module:event-store#event>DeleteFields](#)
- [module:event-store#event:DuplicateFields](#)

Listens to Events:

- [module:event-store#event:ModifyTemplateField](#)
- [module:event-store#event:ModifyMultipleTemplateFields](#)
- [module:event-store#event:DeselectAllTemplateFields](#)

Example

```
var FieldEdit = require('field-edit.js');

// a generic Viewer constructor
function Viewer(opts) {
    var myFieldEdit = FieldEdit(this, {
```

```
        elem: document.getElementById('myFieldEdit')
    });
}
```

`(require("field-edit"))(viewer, options)`

Creates the field editing UI module.

Parameters:

Name	Type	Description						
viewer	Core	The core viewer to which the module will attach.						
options	Object	An options object. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>elem</td><td>HTMLElement</td><td>The element in which the module UI will be inserted.</td></tr></tbody></table>	Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Name	Type	Description						
elem	HTMLElement	The element in which the module UI will be inserted.						

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: field-list

Module: field-list

Manages a list of fields and allows drag and drop reordering.

Fires:

- [module:event-store#event:SelectTemplateField](#)

Listens to Events:

- [module:event-store#event:StateModified](#) for "FieldList" state.

Example

```
var FieldList = require('field-list.js');

// a generic Viewer constructor
function Viewer(opts) {
    var myFieldList = FieldList(this, {
```

```
        elem: document.getElementById('myFieldList')
    });
}
```

`(require("field-list"))(viewer, options)`

Creates the field list UI module.

Parameters:

Name	Type	Description						
viewer	Core	The core viewer to which the module will attach.						
options	Object	An options object. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>elem</td><td>HTMLElement</td><td>The element in which the module UI will be inserted.</td></tr></tbody></table>	Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Name	Type	Description						
elem	HTMLElement	The element in which the module UI will be inserted.						

Methods

`destroy()`

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: fill-checklist

Module: fill-checklist

Displays a list of fields to be completed in the form. As fields are completed, the icon in the checklist item will be updated to reflect the completed state.

Fires:

- [module:event-store#event:ToggleChecklist](#)
- [module:event-store#event:FocusChecklistItem](#)

Listens to Events:

- [module:event-store#event:StateModified](#)

Example

```
var FillChecklist = require('fill-checklist.js');

// a generic Viewer constructor
function Viewer(opts) {
  var myFillChecklist = FillChecklist(this, {
    elem: document.getElementById('myChecklist')
  });
}
```

`(require("fill-checklist"))(viewer, options)`

Creates the checklist UI module.

Parameters:

Name	Type	Description						
viewer	Core	The core viewer to which the module will attach.						
options	Object	An options object. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>elem</td><td>HTMLElement</td><td>The element in which the module UI will be inserted.</td></tr></tbody></table>	Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Name	Type	Description						
elem	HTMLElement	The element in which the module UI will be inserted.						

Methods

`destroy()`

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: fill-form-controller

Module: fill-form-controller

Controls the form. It handles various tasks such as creation of marks, how marks are visually represented, keyboard controls, and field focus management.


```
(require("fill-form-controller"))(viewer)
```

Creates the form controller module.

Parameters:

Name	Type	Description
viewer	Core	The core viewer to which the module will attach.

Fires:

- [module:event-store#event:ModifyState](#)
- [module:event-store#event:FormLoaded](#)
- [module:event-store#event:RegisterKeyCombinations](#)
- [module:event-store#event:CreateDate](#)
- [module:event-store#event:ManageSignatures](#)
- [module:event-store#event:CreateSignature](#)
- [module:event-store#event:Notify](#)
- [module:event-store#event:FormatDate](#)

Listens to Events:

- [module:event-store#event:DisplayForm](#)
- [module:event-store#event:StateModified](#)
- [module:event-store#event:FocusChecklistItem](#)
- [module:event-store#event:KeyCombinationsTriggered](#)
- [module:event-store#event:BurnForm](#)

Example

```
var FillFormController = require('fill-form-  
controller.js');  
  
// a generic Viewer constructor  
function Viewer(opts) {  
    var myFillFormController = FillFormController(this);  
}
```

Methods

[destroy\(\)](#)

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: fill-main-toolbar

Module: fill-main-toolbar

Manages the form's main toolbar.

Fires:

- [module:event-store#event:ModifyState](#)

Listens to Events:

- [module:event-store#event:ToggleChecklist](#)

Example

```
var FillMainToolbar = require('fill-main-toolbar.js');

// a generic Viewer constructor
function Viewer(opts) {
  var myFillMainToolbar = FillMainToolbar(this, {
    elem: document.getElementById('myFillMainToolbar')
  });
}
```

[\(require\("fill-main-toolbar"\)\)\(viewer, options\)](#)

Creates the main toolbar module.

Parameters:

Name	Type	Description						
viewer	Core	The core viewer to which the module will attach.						
options	Object	An options object. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>elem</td><td>HTMLElement</td><td>The element in which the module UI will be inserted.</td></tr></tbody></table>	Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Name	Type	Description						
elem	HTMLElement	The element in which the module UI will be inserted.						

Name	Type	Description
------	------	-------------

Methods

`destroy()`

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: fill-progress

Module: fill-progress

Shows the progress of how many fields have been filled and how many are remaining. If there are required fields, a progress bar indicates the progress of required fields that have been filled. If there are optional fields, text below the progress bar indicates how many optional fields are left.

Listens to Events:

- [module:event-store#event:StateModified](#) for "FieldList" state.

Example

```
var FillProgress = require('fill-progress.js');

// a generic Viewer constructor
function Viewer(opts) {
  var myFillProgress = FillProgress(this, {
    elem: document.getElementById('myFillProgress')
  });
}
```

`(require("fill-progress"))(viewer, options)`

Creates the fill progress UI module.

Parameters:

Name	Type	Description
<code>viewer</code>	Core	The core viewer to which the module will attach.

Name	Type	Description						
options	Object	An options object. Properties						
		<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>elem</td><td>HTMLElement</td><td>The element in which the module UI will be inserted.</td></tr></tbody></table>	Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Name	Type	Description						
elem	HTMLElement	The element in which the module UI will be inserted.						

Methods

destroy()

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: form-controller

Module: form-controller

This module interfaces with `ViewerControl` in order to display the fields being created, as well as update field positioning data. It is in charge of translating between `FieldList` field objects and `ViewerControl` mark objects whenever necessary.

Fires:

- [module:event-store#event:ModifyState](#)
- [module:event-store#event:FormLoaded](#)
- [module:event-store#event:ModifyTemplateField](#)
- [module:event-store#event:ModifyMultipleTemplateFields](#)
- [module:event-store#event:DeselectAllTemplateFields](#)

Listens to Events:

- [module:event-store#event:StateModified](#) for the "FieldList" state.
- [module:event-store#event:DisplayForm](#)
- [module:event-store#event:AlignFields](#)
- [module:event-store#event>DeleteFields](#)

- [module:event-store#event:DuplicateFields](#)
- [module:event-store#event:MatchSizeFields](#)
- [module:event-store#event:ModifyMultipleTemplateFields](#)
- [module:event-store#event:SelectTemplateField](#)
- [module:event-store#event:FormLoaded](#)
- [module:event-store#event:FormCopied](#)
- [module:event-store#event:SaveTemplate](#)
- [module:event-store#event:SaveTemplateCopy](#)

Example

```
var FormController = require('form-controller.js');

// a generic Viewer constructor
function Viewer(opts) {
    var myFormController = FormController(this);
}
```

[\(require\("form-controller"\)\)\(viewer\)](#)

Creates the form controller module.

Parameters:

Name	Type	Description
viewer	Core	The core viewer to which the module will attach.

Methods

[destroy\(\)](#)

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: form-extraction

Module: form-extraction

Performs detection and extraction of form data (such as PDF AcroForm fields) in the document.

When viewer is ready, this module sends a request to determine if the document in a viewing session has acroforms or may contain raster forms. If so, the module provides a modal dialog to cancel or attempt form extraction. The process to get results of the extraction can be cancelled once started. Users are notified of successful conversion and unsupported field types with appropriate popup messages. Errors, occurring during form extraction, are displayed in the same modal dialog, which starts the conversion.

Fires:

- [module:event-store#event:DisplayForm](#)
- [module:event-store#event:Notify](#)
- [module:event-store#event:ModifyState](#)

Example

```
var FormExtraction = require('form-extraction.js');

// a generic Viewer constructor
function Viewer(opts) {
  var myFormExtraction = FormExtraction(this, {
    elem: document.getElementById('formExtraction')
  });
}
```

[\(require\("form-extraction"\)\)\(viewerObj, options\)](#)

Creates the form extraction module.

Parameters:

Name	Type	Description									
viewerObj	Core	The core viewer to which the module will attach.									
options	Object	An options object. <table border="1" data-bbox="427 1664 1321 1769"> <thead> <tr> <th colspan="3">Properties</th> </tr> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>elem</td> <td>HTMLElement</td> <td>The element in which the module UI will be inserted.</td> </tr> </tbody> </table>	Properties			Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Properties											
Name	Type	Description									
elem	HTMLElement	The element in which the module UI will be inserted.									

Module: form-summary

Module: form-summary

This module interfaces with the FieldList state to create a new state. The module will merge groups with fields into a central form summary that can be used by other modules to show the status of the form and the fields/groups that it consists of.

Fires:

- [module:event-store#event:ModifyState](#)

Listens to Events:

- [module:event-store#event:StateModified](#) for the "FieldList" state.

Example

```
var FormSummary = require('form-summary.js');

// a generic Viewer constructor
function Viewer(opts) {
    var myFormSummary = FormSummary(this);
}
```

[\(require\("form-summary"\)\)\(viewer\)](#)

Creates the form summary module.

Parameters:

Name	Type	Description
viewer	Core	The core viewer to which the module will attach.

Methods

[destroy\(\)](#)

Destroys the module.

Module: form-tools

Module: form-tools

Manages the form tools. Selecting a tool determines how the mouse interacts with the document. For example, selecting the Pan tool allows the user to scroll the document by clicking on it and dragging the mouse. Selecting the Signature tool allows the user to use the mouse to create a signature field. After a signature field is added, the Pan tool is automatically selected. The user can click a field tool twice to put it into "sticky" state so that the tool remains selected after adding a field.

Example

```
var FormTools = require('form-tools.js');

// a generic Viewer constructor
function Viewer(opts) {
  var myFormTools = FormTools(this, {
    elem: document.getElementById('myFormTools')
  });
}
```

`(require("form-tools"))(viewer, options)`

Creates the form tools UI module.

Parameters:

Name	Type	Description									
viewer	Core	The core viewer to which the module will attach.									
options	Object	An options object. <table border="1" data-bbox="399 1556 1294 1668"> <thead> <tr> <th colspan="3">Properties</th> </tr> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>elem</td> <td>HTMLElement</td> <td>The element in which the module UI will be inserted.</td> </tr> </tbody> </table>	Properties			Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Properties											
Name	Type	Description									
elem	HTMLElement	The element in which the module UI will be inserted.									

Methods

`destroy()`

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: global-settings-menu

Module: global-settings-menu

Manages global template settings.

Fires:

- [module:event-store#event:ModifyState](#) for "GlobalSettings" and "FieldList" state

Listens to Events:

- [module:event-store#event:AccessGlobalSettings](#)

Example

```
var GlobalSettingsMenu = require('global-settings-menu.js');

// a generic Viewer constructor
function Viewer(opts) {
    var myGlobalSettingsMenu = GlobalSettingsMenu(this, {
        elem:
document.getElementById('myGlobalSettingsMenu')
    });
}
```

`(require("global-settings-menu"))(viewer, options)`

Creates the global settings menu module.

Parameters:

Name	Type	Description						
viewer	Core	The core viewer to which the module will attach.						
options	Object	An options object. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>elem</td><td>HTMLElement</td><td>The element in which the module UI will be inserted.</td></tr></tbody></table>	Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Name	Type	Description						
elem	HTMLElement	The element in which the module UI will be inserted.						

Methods

destroy()

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: global-settings-trigger

Module: global-settings-trigger

triggers .

Example

```
var GlobalSettingsTrigger = require('global-settings-trigger.js');

// a generic Viewer constructor
function Viewer(opts) {
  var myGlobalSettings = GlobalSettingsTrigger(this, {
    elem: document.getElementById('myGlobalSettings')
  });
}
```

`(require("global-settings-trigger"))(viewer, options)`

Creates the global settings trigger UI module.

Parameters:

Name	Type	Description						
viewer	Core	The core viewer to which the module will attach.						
options	Object	An options abject. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>elem</td><td>HTMLElement</td><td>The element in which the module UI will be inserted.</td></tr></tbody></table>	Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Name	Type	Description						
elem	HTMLElement	The element in which the module UI will be inserted.						

Methods

destroy()

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: keyboard-controller

Module: keyboard-controller

Controls the keyboard keys. This module uses [jQuery.hotkeys](#) plugin. If desired it can be replaced with any other keyboard interface code without affecting the keyboard consumer modules.

Fires:

- [module:event-store#event:KeyCombinationsTriggered](#)

Listens to Events:

- [module:event-store#event:RegisterKeyCombinations](#) for "KeyCombinations" state

Example

```
var KeyboardController = require('keyboard-controller.js');

// a generic Viewer constructor
function Viewer(opts) {
    var myKeyboardController = KeyboardController(this);
}
```

[\(require\("keyboard-controller"\)\)\(viewer\)](#)

Creates the keyboard controller module.

Parameters:

Name	Type	Description
viewer	Core	The core viewer to which the module will attach.

Methods

`destroy()`

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: multiple-selection

Module: multiple-selection

Provides UI showing bulk actions to be completed on a selection of more than one field.

Fires:

- [module:event-store#event:AlignFields](#)
- [module:event-store#event>DeleteFields](#)
- [module:event-store#event:DuplicateFields](#)
- [module:event-store#event:MatchSizeFields](#)

Listens to Events:

- [module:event-store#event:ModifyTemplateField](#)
- [module:event-store#event:DeselectAllTemplateFields](#)
- [module:event-store#event:ModifyMultipleTemplateFields](#)

Example

```
var MultipleSelection = require('multiple-selection.js');

// a generic Viewer constructor
function Viewer(opts) {
    var myMultipleSelection = MultipleSelection(this, {
        elem:
document.getElementById('myMultipleSelection')
    });
}
```

`(require("multiple-selection"))(viewer, options)`

Creates the multiple selection UI module.

Parameters:

Name	Type	Description						
viewer	Core	The core viewer to which the module will attach.						
options	Object	An options object. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>elem</td><td>HTMLElement</td><td>The element in which the module UI will be inserted.</td></tr></tbody></table>	Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Name	Type	Description						
elem	HTMLElement	The element in which the module UI will be inserted.						

Methods**destroy()**

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: notification

Module: notification

Displays a notification.

Listens to Events:

- [module:event-store#event:Notify](#)

Example

```
var Notification = require('notification.js');

// a generic Viewer constructor
function Viewer(opts) {
  var myNotification = Notification(this, {
    elem: document.getElementById('myNotification')
  });
}
```

```
(require("notification"))(viewer, options)
```

Creates the notification UI module.

Parameters:

Name	Type	Description									
viewer	Core	The core viewer to which the module will attach.									
options	Object	An options object. <table border="1" data-bbox="400 645 1294 750"> <thead> <tr> <th colspan="3">Properties</th> </tr> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>elem</td> <td>HTMLElement</td> <td>The element in which the module UI will be inserted.</td> </tr> </tbody> </table>	Properties			Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Properties											
Name	Type	Description									
elem	HTMLElement	The element in which the module UI will be inserted.									

Methods

```
destroy()
```

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: page-navigation

Module: page-navigation

Navigates pages.

```
(require("page-navigation"))(viewer, options)
```

Creates the page navigation UI module.

Parameters:

Name	Type	Description									
viewer	Core	The core viewer to which the module will attach.									
options	Object	An options object. <table border="1" data-bbox="400 1980 1294 2085"> <thead> <tr> <th colspan="3">Properties</th> </tr> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>elem</td> <td>HTMLElement</td> <td>The element in which the module UI will be inserted.</td> </tr> </tbody> </table>	Properties			Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Properties											
Name	Type	Description									
elem	HTMLElement	The element in which the module UI will be inserted.									

Name	Type	Description

Example

```
var PageNavigation = require('page-navigation.js');

// a generic Viewer constructor
function Viewer(opts) {
  var myPageNav = PageNavigation(this, {
    elem: document.getElementById('myPageNav')
  });
}
```

Methods

destroy()

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: profile-manager

Module: profile-manager

Provides the ability to create and manage signatures.

`(require("profile-manager"))(viewer, options)`

Created the profile manager module.

Parameters:

Name	Type	Description						
viewer	Core	The core viewer to which the module will attach.						
options	Object	An options object. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>elem</td><td>HTMLElement</td><td>The element in which the module UI will be inserted.</td></tr></tbody></table>	Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Name	Type	Description						
elem	HTMLElement	The element in which the module UI will be inserted.						

Listens to Events:

- [module:event-store#event:CreateSignature](#)
- [module:event-store#event:ManageSignatures](#)

Example

```
var ProfileManager = require('profile-manager.js');

// a generic Viewer constructor
var myProfileManager = ProfileManager(this, {
  elem: document.getElementById('myProfileManager')
});
```

Methods**`destroy()`**

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: state-store

Module: state-store

The state store can keep track of any JSON-style data object for other modules to access and use.

The state store is used as centralized data storage for all modules, especially when concerning data that is shared among 2 or more modules. When individual modules need to update specific data, modifications through the state store ensure that other modules that need to be aware of the latest available data can do so without specific input from the module changing it.

The state store can store any number of states, as defined by a data string. See [module:event-store#event:ModifyState](#). It is able to associate any data object with that particular state, although it is optimized to store key-value Objects.

`(require("state-store"))(viewer)`

Creates and initializes the state store.

Parameters:

Name	Type	Description
viewer	Core	The core viewer to which the module will attach.

Fires:

- [module:event-store#event:StateModified](#)

Listens to Events:

- [module:event-store#event:ModifyState](#)

Example

```
var StateStore = require('state-store.js');

// a generic Viewer constructor
function Viewer(opts) {
    // other modules will expect this to be present
    this.stateStore = StateStore(this);
}
```

Members

(inner) [FieldList](#)

The known set of fields and metadata on the form.

Properties:

Name	Type	Attributes	Default	Description																		
templateDocumentId	string			The unique id used to determine which document b loaded if this value is not defined.																		
formName	string	<optional>	""	The display name of the form.																		
formDefinitionId	string	<optional>		The unique id to use to save the form to the server.																		
formRoles	Object	<optional>		A hash object used to store and access the metadata hash object is the <code>formRoleId</code> of the form role. <table border="1" data-bbox="810 1713 1409 2056"> <thead> <tr> <th colspan="3">Properties</th> </tr> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>formRoleId</td> <td>string</td> <td>The id of the form role.</td> </tr> <tr> <td>displayName</td> <td>string</td> <td>A friendly name for the form.</td> </tr> <tr> <td>fieldColor</td> <td>string</td> <td>The color to use for any field. The format is a pound sign followed by a color code.</td> </tr> <tr> <td>sortIndex</td> <td>number</td> <td>A number representing the sort order of the field.</td> </tr> </tbody> </table>	Properties			Name	Type	Description	formRoleId	string	The id of the form role.	displayName	string	A friendly name for the form.	fieldColor	string	The color to use for any field. The format is a pound sign followed by a color code.	sortIndex	number	A number representing the sort order of the field.
Properties																						
Name	Type	Description																				
formRoleId	string	The id of the form role.																				
displayName	string	A friendly name for the form.																				
fieldColor	string	The color to use for any field. The format is a pound sign followed by a color code.																				
sortIndex	number	A number representing the sort order of the field.																				

Name	Type	Attributes	Default	Description																																								
groups	Object	<optional>		<p>A hash object used to store and access the metadata. The hash object is the <code>groupId</code> of the group.</p> <p>Properties</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Attributes</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>groupId</td> <td>string</td> <td></td> <td>The id of the group.</td> </tr> <tr> <td>displayName</td> <td>string</td> <td></td> <td>A friendly name for the group.</td> </tr> <tr> <td>type</td> <td>string</td> <td></td> <td>The data type of the group. <ul style="list-style-type: none"> checkbox </td> </tr> <tr> <td>data</td> <td>Object</td> <td></td> <td>Data associated with the group. <p>Properties</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Attributes</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>multiple</td> <td>boolean</td> <td></td> <td>Indicates if the group can contain multiple items.</td> </tr> </tbody> </table> </td> </tr> <tr> <td>readOnly</td> <td>boolean</td> <td><optional></td> <td>Indicates whether the group is read-only.</td> </tr> <tr> <td>required</td> <td>boolean</td> <td><optional></td> <td>Indicates whether the group is required.</td> </tr> <tr> <td>formRoleId</td> <td>string</td> <td><optional></td> <td>The form role id of the group.</td> </tr> </tbody> </table>	Name	Type	Attributes	Description	groupId	string		The id of the group.	displayName	string		A friendly name for the group.	type	string		The data type of the group. <ul style="list-style-type: none"> checkbox 	data	Object		Data associated with the group. <p>Properties</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Attributes</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>multiple</td> <td>boolean</td> <td></td> <td>Indicates if the group can contain multiple items.</td> </tr> </tbody> </table>	Name	Type	Attributes	Description	multiple	boolean		Indicates if the group can contain multiple items.	readOnly	boolean	<optional>	Indicates whether the group is read-only.	required	boolean	<optional>	Indicates whether the group is required.	formRoleId	string	<optional>	The form role id of the group.
Name	Type	Attributes	Description																																									
groupId	string		The id of the group.																																									
displayName	string		A friendly name for the group.																																									
type	string		The data type of the group. <ul style="list-style-type: none"> checkbox 																																									
data	Object		Data associated with the group. <p>Properties</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Attributes</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>multiple</td> <td>boolean</td> <td></td> <td>Indicates if the group can contain multiple items.</td> </tr> </tbody> </table>	Name	Type	Attributes	Description	multiple	boolean		Indicates if the group can contain multiple items.																																	
Name	Type	Attributes	Description																																									
multiple	boolean		Indicates if the group can contain multiple items.																																									
readOnly	boolean	<optional>	Indicates whether the group is read-only.																																									
required	boolean	<optional>	Indicates whether the group is required.																																									
formRoleId	string	<optional>	The form role id of the group.																																									
fieldList	Object			<p>A hash object used to store and access the metadata. The hash object is the <code>markId</code> of the viewer mark.</p> <p>Properties</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Attributes</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>markId</td> <td>number</td> <td></td> <td>The mark id of the viewer mark.</td> </tr> <tr> <td>fieldId</td> <td>string</td> <td></td> <td>A unique identifier for the field.</td> </tr> <tr> <td>displayName</td> <td>string</td> <td></td> <td>A friendly name for the field.</td> </tr> <tr> <td>template</td> <td>string</td> <td></td> <td>The template name of the field.</td> </tr> <tr> <td>required</td> <td>boolean</td> <td></td> <td>Indicates whether the field is required.</td> </tr> </tbody> </table>	Name	Type	Attributes	Description	markId	number		The mark id of the viewer mark.	fieldId	string		A unique identifier for the field.	displayName	string		A friendly name for the field.	template	string		The template name of the field.	required	boolean		Indicates whether the field is required.																
Name	Type	Attributes	Description																																									
markId	number		The mark id of the viewer mark.																																									
fieldId	string		A unique identifier for the field.																																									
displayName	string		A friendly name for the field.																																									
template	string		The template name of the field.																																									
required	boolean		Indicates whether the field is required.																																									

Name	Type	Attributes	Default	Description
				readOnly boolean
				horizontalAlignment string
				sortIndex number
				pageNumber number
				pageData Object
				rectangle Object
				fontName string <optional>
				fontColor string <optional>
				multiline boolean <optional>

Name	Type	Attributes	Default	Description																
				<table border="1"> <tr> <td>characterLimit</td> <td>number</td> <td><optional></td> <td>The</td> </tr> <tr> <td>formRoleId</td> <td>string</td> <td><optional></td> <td>The</td> </tr> <tr> <td>groupId</td> <td>string</td> <td><optional></td> <td>The</td> </tr> <tr> <td>defaultValue</td> <td>string Object</td> <td><optional></td> <td>The</td> </tr> </table>	characterLimit	number	<optional>	The	formRoleId	string	<optional>	The	groupId	string	<optional>	The	defaultValue	string Object	<optional>	The
characterLimit	number	<optional>	The																	
formRoleId	string	<optional>	The																	
groupId	string	<optional>	The																	
defaultValue	string Object	<optional>	The																	

Name	Type	Attributes	Default	Description			
							fc
				value	string Object	<optional>	<p>The in t nor val it s 20 dic glc it is "ch obj</p> <p>Pro</p> <p>Na ty</p> <p>va</p>

Name	Type	Attributes	Default	Description				
globalSettings	Object			Settings that apply globally to the template. Properties <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>signatureDateFormat</td> <td>external:"jQuery.fn"~Date</td> </tr> </tbody> </table>	Name	Type	signatureDateFormat	external:"jQuery.fn"~Date
Name	Type							
signatureDateFormat	external:"jQuery.fn"~Date							

Example

```
var fieldList = viewer.stateStore.getState('FieldList');
```

(inner) **FormDefinition**

Defines the schema of the template form that is saved to the server, including all of the metadata required to load and recreate the form.

Properties:

Name	Type	Attributes	Default	Description
templateDocumentId	string			The unique id used to determine which document belongs to the form. The form cannot be loaded if this value is not defined.
formName	string	<optional>	""	The display name of the form.
formDefinitionId	string	<optional>		The unique id to use to save the form to the server.
formRoles	Array	<optional>		The data here is similar to the <code>formRoles</code>

Name	Type	Attributes	Default	Description
				property of <code>module:state-store~FieldList</code> , but represented as an array to be saved to the server. This array will be used to rebuild the <code>formRoles</code> object when a <code>FormDefinition</code> is loaded into the viewer.
<code>groups</code>	Array	<optional>		The data here is similar to the <code>groups</code> property of <code>module:state-store~FieldList</code> , but represented as an array to be saved to the server. This array will be used to rebuild the <code>groups</code> object when a <code>FormDefinition</code> is loaded into the viewer.
<code>formData</code>	Array			The data here is similar to the <code>fieldList</code> property of <code>module:state-store~FieldList</code> , but represented as an array to be saved to the server. This array will be used to rebuild the <code>FieldList</code> object when a <code>FormDefinition</code> is loaded into the viewer. As such, some properties of <code>FieldList.fieldList</code> are excluded when generating the <code>FormDefinition.formData</code> . These exclusions are <code>markId</code> and <code>sortIndex</code> .
<code>globalSettings</code>	Object			An instance of the <code>module:state-store~GlobalSettings</code> object used for this form.

(inner) `PageData`

Defines the set of currently known pages -- ones that have loaded at least once in the viewer -- and their sizes. It is a hash object, using the page number as the object key, and the following properties as the object value.

Properties:

Name	Type	Description
<code>width</code>	number	The width of the page.
<code>height</code>	number	The height of the page.

Methods

`destroy()`

Destroys the instance of the State Store.

`getState(key) → {*|undefined}`

Gets the current state.

Parameters:

Name	Type	Description
key	string	The name of the state value being retrieved.

Returns:

The state value associated with the specified key or `undefined` if the state value does not exist.

Type

* | `undefined`

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: svg-icons

Module: svg-icons

This module appends the icons to the document body. If this module is initialized twice, the icons are not appended since they only need to be appended once.

Members

(inner) `moduleApi`

Properties:

Name	Type	Description
destroy	function	Destroys the module.

Methods

`init()` → `{module:svg-icons~moduleApi}`

Initializes the module. This method will insert the SVG icon sprite into the body of the page. This sprite can be shared between multiple instances of the viewer embedded on the same page.

Returns:

The module API object.

Type

[module:svg-icons~moduleApi](#)

[parseIcons\(dom\)](#)

Parses icons. For any HTML template that contains icons, this method must be called with the HTML template passed as the parameter. Note that the init method must be called before calling `parseIcons`.

Parameters:

Name	Type	Description
dom	HTMLElement	A parent DOM element, or jQuery-wrapped element, that contains the icons that need to be parsed.

(inner) [module:svg-icons#updateIcon](#) Parses a single icon.(elem)

Parameters:

Name	Type	Description
elem	HTMLElement	The icon that needs to be parsed.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: [template-io](#)

Module: [template-io](#)

Manages the saving and loading of template files.

Fires:

- [module:event-store#event:ModifyState](#)
- [module:event-store#event:DisplayForm](#)
- [module:event-store#event:FormCopied](#)
- [module:event-store#event:TemplateSaved](#)
- [module:event-store#event:TemplateSaveFailed](#)

Listens to Events:

- [module:event-store#event:SaveTemplate](#)
- [module:event-store#event:SaveTemplateCopy](#)

Example

```
var TemplateIO = require('template-io.js');

// a generic Viewer constructor
function Viewer(opts) {
    var myTemplateIO = TemplateIO(this);
}
```

[\(require\("template-io"\)\)\(viewer\)](#)

Creates the template IO module.

Parameters:

Name	Type	Description
viewer	Core	The core viewer to which the module will attach.

Methods

[destroy\(\)](#)

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: [template-manager](#)

Module: [template-manager](#)

Provides a UI to name and save templates.

[\(require\("template-manager"\)\)\(viewer, options\)](#)

Creates the template manager module.

Parameters:

Name	Type	Description						
viewer	Core	The core viewer to which the module will attach.						
options	Object	An options object. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>elem</td><td>HTMLElement</td><td>The element in which the module UI will be inserted.</td></tr></tbody></table>	Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Name	Type	Description						
elem	HTMLElement	The element in which the module UI will be inserted.						

Fires:

- [module:event-store#event:ModifyState](#)
- [module:event-store#event:SaveTemplate](#)
- [module:event-store#event:Notify](#)

Listens to Events:

- [module:event-store#event:FormLoaded](#)
- [module:event-store#event:FormCopied](#)
- [module:event-store#event:TemplateSaved](#)
- [module:event-store#event:TemplateSaveFailed](#)

Example

```
var TemplateManager = require('template-manager.js');

// a generic Viewer constructor
function Viewer(opts) {
  var myTemplateManager = TemplateManager(this, {
    elem: document.getElementById('myTemplateManager')
  });
}
```

Methods**destroy()**

Destroys the module.

Module: template-name-header

Module: template-name-header

Provides the ability to display the currently loaded template name as a header.

Listens to Events:

- [module:event-store#event:DisplayForm](#)

Example

```
var TemplateNameHeader = require('template-name-
header.js');

// a generic Viewer constructor
var myTemplateNameHeader = TemplateNameHeader(this, {
  elem: document.getElementById('myTemplateNameHeader')
});
```

`(require("template-name-header"))(viewer, options)`

Created the template name header module.

Parameters:

Name	Type	Description						
viewer	Core	The core viewer to which the module will attach.						
options	Object	An options object. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>elem</td><td>HTMLElement</td><td>The element in which the module UI will be inserted.</td></tr></tbody></table>	Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Name	Type	Description						
elem	HTMLElement	The element in which the module UI will be inserted.						

Methods

`destroy()`

Destroys the module.

Module: text-input

Module: text-input

A text input.

Examples

```
<!--The following HTML includes a text input component.  
An element is specified as a text input by setting the  
data-pcc-component attribute  
to "textinput".-->  
<div data-pcc-component="textinput" data-pcc-  
name="displayName" class="pcc-textbox"></div>
```

```
// Require the text input module.  
var TextInput = require('../elements/text-input.js');  
  
// Pass each text input element to the text input module to  
initialize each text input.  
// parent is the element that contains the text input  
element.  
$(parent).find('[data-pcc-  
component="textinput"]').each(function() {  
    TextInput(this);  
});
```

(require("text-input"))(e1) → {HTMLElement}

Parses and initializes a text input.

Parameters:

Name	Type	Description
e1	HTMLElement	The parent element in which to parse for the text input component.

Returns:

The parsed text input element.

Type

HTMLElement

Members

off :module:event-store~off

Removes an event handler from the text input.

Type:

- module:event-store~off

on :module:event-store~on

Registers an event handler on the text input.

Type:

- module:event-store~on

Methods

destroy()

Destroys the text input component.

focus()

Focuses the text input component

hideError() → {HTMLElement}

Removes the text below the text input if showError was called previously to show text below the text input.

Returns:

The text input element.

Type

HTMLElement

showError(error) → {HTMLElement}

Shows the specified text below the text input.

Parameters:

Name	Type	Description
error	string	The error text to show below the text input.

Returns:

The text input element.

Type
HTMLElement

`value(text) → {Object}`

Gets or sets the value of the text input.

Parameters:

Name	Type	Description
text	string	The text to show in the text input.

Returns:

The text input element if a value is passed. Otherwise, the current value is returned.

Type
Object

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Module: zoom-fit

Module: zoom-fit

Zooms and fits the document. This module will allow the user to zoom in and out, set a specific scale factor, or set a page fit mode that will be maintained when the browser window is resized.

Example

```
var ZoomFit = require('zoom-fit.js');  
  
// a generic Viewer constructor
```

```
function Viewer(opts) {  
    var myZoomFit = ZoomFit(this, {  
        elem: document.getElementById('myZoomFit')  
    });  
}
```

`(require("zoom-fit"))(viewer, options)`

Creates the zoom and fit module.

Parameters:

Name	Type	Description						
viewer	Core	The core viewer to which the module will attach.						
options	Object	An options object. Properties <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>elem</td><td>HTMLElement</td><td>The element in which the module UI will be inserted.</td></tr></tbody></table>	Name	Type	Description	elem	HTMLElement	The element in which the module UI will be inserted.
Name	Type	Description						
elem	HTMLElement	The element in which the module UI will be inserted.						

Methods

`destroy()`

Destroys the module.

Documentation generated by [JSDoc 3.5.5](#) on Mon Jun 14 2021 12:02:27 GMT-0400 (Eastern Daylight Time)

Cloud Authentication

Overview

This section contains the following information on the PrizmDoc Cloud specific APIs:

- [Authenticating Requests](#)
- [OAuth](#)

NOTE: The APIs in this section are for PrizmDoc Cloud users only and do not apply to PrizmDoc Viewer Self-Hosted users.

Authenticating Requests

Introduction

When using PrizmDoc Cloud, you must authenticate all HTTP requests. You can do this in one of two different ways:

- [Using Your API Key](#)
- [Using OAuth](#)

Using Your API Key

Include an `acs-api-key` header with your API key as the value.

Example

```
POST https://api.accusoft.com/PCCIS/V1/ViewingSession acs-api-key: <your key here>
Content-Type: application/json
{
  "render": {
    "html5": {
      "alwaysUseRaster": false
    }
  }
}
```

To obtain an API key, visit <https://www.accusoft.com/portal/>.

Using OAuth

To facilitate granular access to PrizmDoc Cloud account data, especially document storage, another level of authorization must be implemented to further identify requests. The OAuth 2.0 specification allows for this type of authorization. Using OAuth, PrizmDoc Cloud can not only authorize a request via the PrizmDoc Cloud API key, but also by custom user information provided by clients. Ultimately, this will allow for the creation of rules by PrizmDoc Cloud customers to limit access to various resources.

OAuth Authorization Method

Per the OAuth spec, PrizmDoc Cloud implements the client credentials grant method. This relies on authentication through the PrizmDoc Cloud API key. See: <https://www.rfc-editor.org/rfc/rfc6749.html#section-4.4>. By default, the expire time for the access token is set to one day. Initially at least, the use of OAuth necessitates the use of server-side code for the creation of the access tokens. For more information, see the [OAuth API](#) reference.

OAuth

OAuth

Using OAuth, PrizmDoc Cloud can not only authorize a request via the PrizmDoc Cloud API key, but also by custom user information. Ultimately, this will allow you to create rules to limit access to various resources.

Available URLs

URL	Description
POST /v1/authTokens	Retrieves an authorization token that can be used to authenticate calls to PrizmDoc Cloud Services.

POST /v1/authTokens

Retrieves an authorization token that can be used to authenticate calls to PrizmDoc Cloud Services.

Request

Request Headers

Name	Description
<code>acs-api-key</code>	Required {{api-key}}
<code>Content-Type</code>	Required application/x-www-form-urlencoded

Request Body

```
grant_type=client_credentials&scope={client customer user id} {client custom role}
```

Successful Response

Response Body

Success returns code 200 - OK

- `Content-Type`: application/json
- `Cache-Control`: no-store
- `Pragma`: no-cache

Error Responses

Status Code	JSON <code>errorCode</code>	Description
400	"Bad Request"	Returned when data is incorrect within body of the request.
401	"Unauthorized"	Returned when the key provided in the <code>acs-api-key</code> is incorrect or missing.

Examples

Request

```
grant_type=client_credentials&scope={client customer user id} {client custom role}
```

```
grant_type=client_credentials&scope=userid:1234 role:admin
```

Response

Successful Response

```
{
  "access_token": "{valid access token}",
  "token_type": "acs-oauth",
  "expires_in": {time in seconds},
  "scope": {requested scope} // user:{useridValue} role:{roleValue}
}
```

```
{
  "access_token": "PrizmDoc Cloud-Hosted-ACCESS-TOKEN",
  "token_type": "acs-oauth",
  "expires_in": 3600,
  "scope": "userid:1234 role:admin"
}
```

Error Response

```
{
  "error": "invalid_request" // error code definitions: https://www.rfc-
editor.org/rfc/rfc6749.html#section-5.2
}
```

PAS REST API

Overview

The PAS REST API supports **viewing functionality**.

For automated document processing that does not involve a viewer, use the [PrizmDoc Server REST API](#) instead.

General Information

- [Base URL for PAS](#)
- [API Data Types](#)
- [Back-end Proxy](#)

Areas of Functionality

The PAS REST APIs can be broken into three main groups:

Application Development

These are the REST APIs you will most-commonly use:

- [Viewing Sessions](#)
- [Viewing Package Creators](#)
- [Viewing Packages](#)

Self-Hosted Administration

This REST API is useful if you are self-hosting PAS instances:

- [Health](#)

Viewer Support

These REST APIs are used by our viewer. It is uncommon for your application to need to use them:

- [Attachments](#)
- [Content Converters](#)
- [Content Converters \(Deprecated\)](#)
- [Form Definitions](#)
- [Form Extractors](#)
- [Image Stamps](#)
- [Legacy Create Session](#)
- [Markup Burners](#)
- [Markup Layers](#)
- [Markup XML](#)
- [Search Tasks](#)

General Information

This section contains the following information:

- [Base URL for PAS](#)
- [API Data Types](#)
- [Back-end Proxy](#)

Base URL for PAS

Introduction

When making REST API calls to PAS, you need to use the appropriate base URL.

PrizmDoc Cloud PAS

If you are using the Accusoft-managed PAS which is part of PrizmDoc Cloud, the base URL for the PAS REST APIs varies depending on your region:

Region	PAS Base URL
United States	https://api.accusoft.com/prizmdoc

Note the trailing `/prizmdoc` at the end of the base URL. When using the Accusoft-managed PAS which is part of PrizmDoc Cloud, this is a required part of the base URL.

Remember that PrizmDoc Cloud requires you to authenticate each request with an `acs-api-key` request header.

Self-Hosted PAS

When hosting PAS yourself, just use the hostname and port of your PAS instance (or the hostname and port of the load balancer which sits in front of your PAS cluster).

For a typical installation on localhost, the PAS base URL will be:

```
http://localhost:3000
```

API Data Types

Introduction

Prizm Application Services REST API uses a data type system that is slightly more detailed and more specific than JavaScript's common data types (integer, date, and dateTime). These data types are used for defining properties of the JSON objects in the body of the POST requests and in the body of the responses where applicable.

The table below shows the supported data types by the API:

Type	Description	Example
number	Any number. This includes numbers with or without decimals.	1000.15 or 1500
integer	whole numbers only	120
boolean	true or false (without quotes)	true or false
date	This is the ISO 8601 profile for the full-date as described in the RFC 3339 section 5.6, Internet Date/Time Format . The syntax for full-date as described in this document is as <code>full-date = YYYY(4 digits) "-" MM(01 through 12) "-" " DD(01 through 31)</code>	2015-05-12
dateTime	This is the ISO 8601	November 5, 2015, 8:15:30 am, US Eastern Standard Time : 2015-11-

Type	Description	Example
	<p>profile for the date-time as described in the RFC 3339 section 5.6, Internet Date/Time Format. The date-time syntax described in this document is</p> <pre>date-time = YYYY "-" MM "-" " DD "T" hh(00 through 23) ":" mm(00 through 59) ":" ss(00 through 59) "Z" / ("+" / "-") hh(00 through 23) ":" mm(00 through 59). This profile defines two ways of handling time zone offsets: <ol style="list-style-type: none"> 1. Times are expressed in UTC (Coordinated Universal Time), with a special UTC designator ("Z"). 2. Times are expressed in local time, together with a time zone offset in hours and minutes. A time zone offset of "+hh:mm" indicates that the date/time uses a local </pre>	<pre>05T08:15:30-05:00 Same instant in UTC : 2015-11-05T13:15:30Z</pre>

Type	Description	Example
	time zone which is "hh" hours and "mm" minutes ahead of UTC. A time zone offset of "-hh:mm" indicates that the date/time uses a local time zone which is "hh" hours and "mm" minutes behind UTC.	
object	A JSON object	<code>{"fileName": "sample.doc"}</code>
array	An array object	<code>["one", "two", "three"]</code>
string	A sequence of zero or more characters	<code>"abcdefhh"</code>
url	A string which is a URL	<code>"http://example.com"</code>
urlSafeBase64	A URL-safe base64 encoded string, according to RFC 4648 Section 5	<code>"Pqu_fKOCYd1QM5oJW6pz-suKQ-2fuxbdZtCKcApvMFVP9GGKv99crwyXTr6AZjrC5vvi3acnZVLgyEXzA"</code>

Back-end Proxy

Introduction

The following routes are proxied through PAS to PrizmDoc Server. Many of them are direct proxies to corresponding PrizmDoc Server routes, and more information about them can be found in the [PrizmDoc Server REST API documentation](#).

Routes for document viewing

GET /Document/q/Attributes?DocumentID=u{viewingSessionId}

Routes key: `GetDocumentAttributes`

Gets a page count for the source document of a viewing session.

For more information, see the PrizmDoc Server endpoint [GET /PCCIS/V1/Document/q/Attributes](#).

GET /Document/q/{PageNumberBegin}-{PageNumberEnd}/Text?DocumentID=u{viewingSessionId}

Routes key: `GetPageText`

Gets currently-available text and text metadata for a range of pages for the source document of a viewing session.

For more information, see the PrizmDoc Server endpoint [GET /PCCIS/V1/Document/q/{PageNumberBegin}-{PageNumberEnd}/Text](#).

GET /v2/viewingSessions/{viewingSessionId}/revisionData?limit={limit}&continueToken={continueToken}

Routes key: `GetDocumentRevisionData`

Gets objects which describe known changes between the two documents used as input to a comparison viewing session.

For more information, see the PrizmDoc Server endpoint [GET /v2/viewingSessions/{viewingSessionId}/revisionData](#).

GET /Page/q/{pageNumber}?DocumentID=u{viewingSessionId}

Routes key: `GetPage`

Gets SVG or an image for a page of the source document of a viewing session.

For more information, see the PrizmDoc Server endpoint [GET /PCCIS/V1/Page/q/{PageNumber}](#).

GET /Page/q/{pageNumber}/Tile/{x}/{y}/{width}/{height}?DocumentID=u{viewingSessionId}

Routes key: `GetPageTile`

Gets a "tile" image, a part of a page, for a page of the source document of a viewing session.

For more information, see the PrizmDoc Server endpoint [GET /PCCIS/V1/Page/q/{PageNumber}/Tile/{x}/{y}/{width}/{height}](#).

GET /Page/q/{pageNumber}/Attributes?DocumentID=u{viewingSessionId}

Routes key: `GetPageAttributes`

Gets metadata for a page of the source document of a viewing session.

For more information, see the PrizmDoc Server endpoint [GET /PCCIS/V1/Page/q/{PageNumber}/Attributes](#).

GET /Page/q/{pageNumber}/{width}x{height}?DocumentID=u{viewingSessionId}

Routes key: `GetThumbnail`

Gets a thumbnail image for a page of the source document of a viewing session.

For more information, see the PrizmDoc Server endpoint [GET /PCCIS/V1/Page/q/{PageNumber}/{Width}x{Height}](#).

Routes related to the original document

POST /ViewingSession/u{viewingSessionId}/Replacement

Routes key: `CreateViewingSessionReplacement`

Replaces the existing viewing session with a new one. Useful for supplying passwords on password protected documents.

Example

```
POST pas_base_url/ViewingSession/uXYZ.../Replacement
Content-Type: application/json

{
  "password": "pdfPassword"
}
```

GET /ViewingSession/u{viewingSessionId}/SourceFile

Routes key: `GetSourceFile`

Downloads the original document that is being viewed.

For more information, see the PrizmDoc Server endpoint [GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/SourceFile](#).

GET /SaveDocument/q?DocumentID=u{viewingSessionId}

Routes key: `SaveDocument`

Downloads the original document that is being viewed, automatically setting the `Content-Disposition` header with a filename based upon information given in the original `POST /ViewingSession` request. Like [GET /ViewingSession/u{viewingSessionId}/SourceFile](#), except that this endpoint does not allow you to provide a custom `ContentDispositionFilename`.

This endpoint is designed for requests which originate from a browser. If you are writing server-side application code which downloads the document from a viewing session, consider using [GET /ViewingSession/u{viewingSessionId}/SourceFile](#) instead.

Example

```
GET pas_base_url/SaveDocument/q?DocumentID=uXYZ...
```

GET /v2/viewingSessions/{ViewingSessionID}/sourceFile/original

Routes key: `GetOriginalSourceFile`

Downloads the "original" (as opposed to "revised") document that is being used for a comparison session.

For more information, see the PrizmDoc Server endpoint [GET /v2/viewingSessions/{viewingSessionId}/sourceFile/original](#).

GET /v2/viewingSessions/{ViewingSessionID}/sourceFile/revised

Routes key: `GetRevisedSourceFile`

Downloads the "revised" document that is being used for a comparison session.

For more information, see the PrizmDoc Server endpoint [GET /v2/viewingSessions/{viewingSessionId}/sourceFile/revised](#).

GET /ViewingSession/u{viewingSessionId}/Attachments

Routes key: `GetAttachments`

Gets information about the document attachments (such as ones available on an `eml` or `msg` file).

For more information, see the PrizmDoc Server endpoint [GET /PCCIS/V1/ViewingSession/u{ViewingSessionID}/Attachments](#).

Routes for markup burning

POST /ViewingSession/u{viewingSessionId}/MarkupBurner

Routes key: `CreateMarkupBurner`

Creates a document burning task for the specific document in the viewing session.

For more information, see the PrizmDoc Server endpoint [POST /PCCIS/V1/ViewingSession/u{viewingSessionId}/MarkupBurner](#).

GET /ViewingSession/u{viewingSessionId}/MarkupBurner/{markupBurnerId}

Routes key: `PollMarkupBurner`

Checks the status of the markup burning task.

For more information, see the PrizmDoc Server endpoint [GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/MarkupBurner/{processId}](#).

GET /ViewingSession/u{viewingSessionId}/MarkupBurner/{markupBurnerId}/Document

Routes key: `GetBurnedDocument`

Downloads the resulting burned-in document.

For more information, see the PrizmDoc Server endpoint [GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/MarkupBurner/{processId}/Document](#).

GET /License/ClientViewer

Routes key: `GetClientViewerLicense`

NOTE: This URL has been deprecated and will be removed from the public API in a future release. It no longer functions and returns HTTP 500 `Internal Server Error`.

Application Development

These are REST APIs you will most-commonly use:

- [Viewing Sessions](#)
- [Viewing Package Creators](#)
- [Viewing Packages](#)

Viewing Sessions

Introduction

The *viewing sessions* REST API is used by your application every time you need to show a document with our

viewer.

A *viewing session* is a temporary resource. At a high level, it takes a source document as input and then, while the viewing session exists, it is able to produce document content in a form that can be displayed in the browser. Creating a viewing session ([POST /ViewingSession](#)) is very fast. Any document conversion work necessary will happen in the background *after* the viewing session has been created.

Our viewer, running in the browser, always makes requests for document content via an existing viewing session. So, to render HTML containing our viewer, your application must first create a viewing session. For more information about this, see [Getting Started with Document Viewing](#).

Available URLs

URL	Description
POST /ViewingSession	Creates a viewing session.
PUT /ViewingSession/u{viewingSessionId}/SourceFile	Uploads a document to the session.
PUT /v2/viewingSessions/{viewingSessionId}/sourceFile/original	Uploads original document to the comparison viewing session.
PUT /v2/viewingSessions/{viewingSessionId}/sourceFile/revised	Uploads revised document to the comparison viewing session.
GET /ViewingSession/u{viewingSessionId}/SourceFile	Downloads the source document in use for a viewing session.
GET /v2/viewingSessions/{viewingSessionId}/sourceFile/original	When viewing a comparison of two documents, downloads the first of the two source documents.
GET /v2/viewingSessions/{viewingSessionId}/sourceFile/revised	When viewing a comparison of two documents, downloads the second of the two source documents.
GET /v2/viewingSessions/{viewingSessionId}/restrictions	Returns information about any restrictions enforced by the server for the current viewing session.
POST /ViewingSession/u{viewingSessionId}/Replacement	Replaces a viewing session with new parameters.

POST /ViewingSession

Routes key: `PostViewingSession`

Creates a new viewing session. At a high level, a viewing session takes a source document as input and produces HTML page content and document text as output.

Request

Request Headers

Name	Description
<code>Content-Type</code>	Should be <code>application/json</code>

Request Body

- `source` (object) Object describing a source document where it comes from, such as local documents,

documents specified via URL, etc. Properties relevant to PrizmApplicationService will reside in this object.

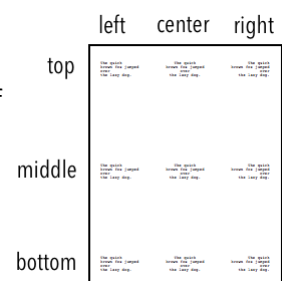
- When **uploading** the source document in a subsequent request:
 - `type` (string) **Required.** Set to `upload`.
 - `displayName` (string) **Required.** Unique filename, including the extension, for the document that will be uploaded.
 - **IMPORTANT:** You must use a value which is specific for the document you are going to upload. For optimal performance, you should NEVER use the same value for two different documents. We use this value internally (when running in clustered mode) to determine which machine in the cluster this viewing session should be assigned to. This allows us to ensure that viewing sessions with the same `displayName` can be consistently assigned to the same machine in the cluster, taking advantage of previously cached conversion output. But we are assuming you will provide us a value which is unique for the document which you are about to upload. If, instead, you simply use a hard-coded value for all documents, you will mistakenly force all work to a single machine in the cluster. Make sure this value is unique for the document you are about to upload!
 - `markupId` (string) The ID to use when querying markup with the XML and JSON layer APIs. If one isn't passed, we create one from the other information we're given. The `markupId` is required to have a non-zero length, should be less than 256 characters, and should consist of characters defined by [RFC4648 - Base 64 Encoding with URL and Filename Safe Alphabet](#).
 - `downloadName` (string) Filename a browser should use when an end user downloads the source document (the filename of the `Content-Disposition` response header). We will use the `displayName` if this option is not provided.
 - `fileExtension` (string) A file extension that's used if the File Detection Service cannot determine what type of file was uploaded. We will get the extension from `displayName` if this option is not provided. This parameter only accepts alpha-numeric characters.
 - `documentId` (string) A customer supplied identifier which is used to uniquely identify a viewing package. The `documentId` is required to have a non-zero length less than 256 characters and must consist of characters defined by [RFC4648 - Base 64 Encoding with URL and Filename Safe Alphabet](#). If a viewing package exists for the given `documentId`, PAS will use the content stored in that package and avoid sending any work to PrizmDoc Server. If a viewing package does *not* exist for the given `documentId`, PAS will ask PrizmDoc Server to immediately start preparing content for this viewing session *and* will also start creating a new viewing package in the background. If an errored viewing package exists for the given `documentId`, PAS will effectively ignore the `documentId` and ask PrizmDoc Server to dynamically prepare content for the viewing session.
- When using a **URL** for the source document:
 - `type` (string) **Required.** Set to `url`.
 - `url` (string) **Required.** URL to create a viewing session from.
 - `headers` (object) Request headers to send from PrizmApplicationServices when retrieving the URL.
 - `acceptBadSslCertificate` (boolean) If true, requires SSL certificates be valid. Default is `false`.
 - `markupId` (string) The ID to use when querying markup with the XML and JSON layer APIs. If one isn't passed, we create one from the other information we're given. The `markupId` is required to have a non-zero length, should be less than 256 characters, and should consist of characters defined by [RFC4648 - Base 64 Encoding with URL and Filename Safe Alphabet](#).
 - `downloadName` (string) Filename a browser should use when an end user downloads the source document (the filename of the `Content-Disposition` response header). We will use the `displayName` if this option is not provided.

- `fileExtension` (string) A file extension that's used if the File Detection Service cannot determine what type of file was uploaded. This parameter only accepts alpha-numeric characters.
- `documentId` (string) A customer supplied identifier which is used to uniquely identify a viewing package. The `documentId` is required to have a non-zero length less than 256 characters and must consist of characters defined by [RFC4648 - Base 64 Encoding with URL and Filename Safe Alphabet](#). If a viewing package exists for the given `documentId`, PAS will use the content stored in that package and avoid sending any work to PrizmDoc Server. If a viewing package does *not* exist for the given `documentId`, PAS will ask PrizmDoc Server to immediately start preparing content for this viewing session *and* will also start creating a new viewing package in the background. If an errored viewing package exists for the given `documentId`, PAS will effectively ignore the `documentId` and ask PrizmDoc Server to dynamically prepare content for the viewing session.
- When using a **local file on the PAS server** as the source document (self-hosted only, not recommended):
 - `type` (string) **Required.** Set to `document`.
 - `fileName` (string) **Required.** Filename that the Storage Provider API can use.
 - `markupId` (string) The ID to use when querying markup with the XML and JSON layer APIs. If one isn't passed, we create one from the other information we're given. The `markupId` is required to have a non-zero length, should be less than 256 characters, and should consist of characters defined by [RFC4648 - Base 64 Encoding with URL and Filename Safe Alphabet](#).
 - `downloadName` (string) Filename a browser should use when an end user downloads the source document (the filename of the `Content-Disposition` response header). We will use the `displayName` if this option is not provided.
 - `fileExtension` (string) A file extension that's used if the File Detection Service cannot determine what type of file was uploaded. We will get the extension from `fileName` if this option is not provided. This parameter only accepts alpha-numeric characters.
 - `documentId` (string) A customer supplied identifier which is used to uniquely identify a viewing package. The `documentId` is required to have a non-zero length less than 256 characters and must consist of characters defined by [RFC4648 - Base 64 Encoding with URL and Filename Safe Alphabet](#). If a viewing package exists for the given `documentId`, PAS will use the content stored in that package and avoid sending any work to PrizmDoc Server. If a viewing package does *not* exist for the given `documentId`, PAS will ask PrizmDoc Server to immediately start preparing content for this viewing session *and* will also start creating a new viewing package in the background. If an errored viewing package exists for the given `documentId`, PAS will effectively ignore the `documentId` and ask PrizmDoc Server to dynamically prepare content for the viewing session.
- When using a **viewing package** with pre-converted content:
 - `type` (string) **Required.** Set to `viewingPackage`.
 - `documentId` (string) **Required.** A customer supplied identifier which is used to uniquely identify a viewing package. The `documentId` is required to have a non-zero length less than 256 characters and must consist of characters defined by [RFC4648 - Base 64 Encoding with URL and Filename Safe Alphabet](#). This value must refer to a completed viewing package, otherwise the viewing session will not be created and an error will be returned.
 - `markupId` (string) The ID to use when querying markup with the XML and JSON layer APIs. If one isn't passed, we create one from the other information we're given. The `markupId` is required to have a non-zero length, should be less than 256 characters, and should consist of characters defined by [RFC4648 - Base 64 Encoding with URL and Filename Safe Alphabet](#).
 - `downloadName` (string) Filename a browser should use when an end user downloads the source document (the filename of the `Content-Disposition` response header). We will use the `displayName` if this option is not provided.
- When **comparing** two Microsoft Word documents:
 - `type` (string) **Required.** Set to `comparison`.

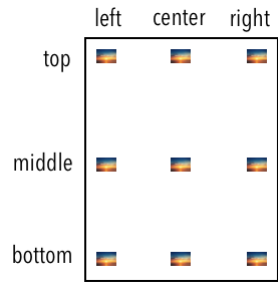
- `original` (object) **Required.** Original document to compare.
 - When **uploading** the original document in a subsequent request: (**NOTE:** Use [PUT /v2/viewingSessions/{viewingSessionId}/sourceFile/original](#) and/or [PUT /v2/viewingSessions/{viewingSessionId}/sourceFile/revised](#) to upload the original and/or revised documents):
 - `type` (string) **Required.** Set to `upload`.
 - `displayName` (string) **Required.** Unique filename, including the extension, for the document that will be uploaded.
 - `downloadName` (string) Filename a browser should use when an end user downloads the original document (the filename of the `Content-Disposition` response header). We will use the `displayName` if this option is not provided.
 - `fileExtension` (string) A file extension that's used if the File Detection Service cannot determine what type of file was uploaded. This parameter only accepts alpha-numeric characters. We will get the extension from `displayName` if this option is not provided.
 - When using a **URL** for the original document:
 - `type` (string) **Required.** Set to `url`.
 - `url` (string) **Required.** URL to create a viewing session from.
 - `headers` (object) Request headers to send from PrizmApplicationServices when retrieving the URL.
 - `acceptBadSslCertificate` (boolean) If true, requires SSL certificates be valid. Default is `false`. > **NOTE:** To use your own certificate authority, you need to specify an agent that was created with that CA as an option.
 - `downloadName` (string) Filename a browser should use when an end user downloads the original document (the filename of the `Content-Disposition` response header). We will use the `displayName` if this option is not provided.
 - `fileExtension` (string) A file extension that's used if the File Detection Service cannot determine what type of file was uploaded. This parameter only accepts alpha-numeric characters.
 - When using a **local file on the PAS server** as the original document (self-hosted only, not recommended):
 - `type` (string) **Required.** Set to `document`.
 - `fileName` (string) **Required.** Filename that the Storage Provider API can use.
 - `downloadName` (string) Filename a browser should use when an end user downloads the original document (the filename of the `Content-Disposition` response header). We will use the `displayName` if this option is not provided.
 - `fileExtension` (string) A file extension that's used if the File Detection Service cannot determine what type of file was uploaded. This parameter only accepts alpha-numeric characters. We will get the extension from `fileName` if this option is not provided.
- `revised` (object) **Required.** Revised document to compare.
 - When **uploading** the revised document in a subsequent request: (**NOTE:** Use [PUT /v2/viewingSessions/{viewingSessionId}/sourceFile/original](#) and/or [PUT /v2/viewingSessions/{viewingSessionId}/sourceFile/revised](#) to upload the original and/or revised documents):
 - `type` (string) **Required.** Set to `upload`.
 - `displayName` (string) **Required.** Unique filename, including the extension, for the document that will be uploaded.
 - `downloadName` (string) Filename a browser should use when an end user downloads the revised document (the filename of the `Content-Disposition`

- response header). We will use the `displayName` if this option is not provided.
- `fileExtension` (string) A file extension that's used if the File Detection Service cannot determine what type of file was uploaded. This parameter only accepts alpha-numeric characters. We will get the extension from `displayName` if this option is not provided.
 - *When using a **URL** for the revised document:*
 - `type` (string) **Required.** Set to `url`.
 - `url` (string) **Required.** URL to create a viewing session from.
 - `headers` (object) Request headers to send from PrizmApplicationServices when retrieving the URL.
 - `acceptBadSslCertificate` (boolean) If true, requires SSL certificates be valid. Default is `false`. > **NOTE:** *To use your own certificate authority, you need to specify an agent that was created with that CA as an option.*
 - `downloadName` (string) Filename a browser should use when an end user downloads the revised document (the filename of the `Content-Disposition` response header). We will use the `displayName` if this option is not provided.
 - `fileExtension` (string) A file extension that's used if the File Detection Service cannot determine what type of file was uploaded. This parameter only accepts alpha-numeric characters.
 - *When using a **local file on the PAS server** as the revised document (self-hosted only, not recommended):*
 - `type` (string) **Required.** Set to `document`.
 - `fileName` (string) **Required.** Filename that the Storage Provider API can use.
 - `downloadName` (string) Filename a browser should use when an end user downloads the revised document (the filename of the `Content-Disposition` response header). We will use the `displayName` if this option is not provided.
 - `fileExtension` (string) A file extension that's used if the File Detection Service cannot determine what type of file was uploaded. This parameter only accepts alpha-numeric characters. We will get the extension from `fileName` if this option is not provided.
 - `markupId` (string) The ID to use when querying markup with the XML and JSON layer APIs. If one isn't passed, we create one from the other information we're given. The `markupId` is required to have a non-zero length, should be less than 256 characters, and should consist of characters defined by [RFC4648 - Base 64 Encoding with URL and Filename Safe Alphabet](#).
 - `downloadName` (string) Filename a browser should use when an end user downloads the source document (the filename of the `Content-Disposition` response header). We will use the `displayName` if this option is not provided.
 - `documentExtension` (String) - File extension of the source document (like `"docx"`, `"html"`, or `"csv"`) used to indicate the source document file format. Often unnecessary. Only required when 1) the `documentSource` is `"http"` or `"file"` but the `externalId` containing the URL or file path did not end with a recognizable file extension and 2) the source document file format could not be automatically detected (this most-commonly occurs for text-based file formats, such as `txt`, `csv`, and `html`). > **NOTE:** *If Format Detection is disabled, then the source document format will never be automatically detected.*
 - `password` (String) Password to use when opening a password-protected source document.
 - `tenantId` (String) Custom, arbitrary tenant id to be associated with the viewing session. PrizmDoc Server has no concept of tenants; if provided, this metadata is only for use by the calling application.
 - `origin` (Object) Custom, arbitrary set of key/value string pairs to be associated with the viewing session. Intended for associating end user origin data (like IP address or hostname), but you can use any set of key/value strings you want.
 - `render` (Object) Options which control how browser content is rendered:

- `html5` (Object) Options when the output format is HTML:
 - `alwaysUseRaster` (Boolean) **Required.** Determines whether only raster data, instead of SVG, should be created for the viewing session. With modern browsers, it is rare to only want raster. This is typically set to `false`.
 - `svgMaxImageSize` (Number) The maximum edge length, in pixels, that is allowable for any image when creating SVG. For example, a value of `8000` would ensure that any images in a PDF whose width or height were greater than `8000` pixels would be down-sampled before the image was added to the final SVG. Default is configurable, but is typically `8000`. To disable this optimization, use a value of `0`.
 - `vectorTolerance` (Number) For CAD documents, the amount of path simplification that is allowable when creating the SVG. Path simplification will merge points which are "close together" to create optimized SVG. You can think of this value as defining what "close together" means. Higher values introduce more simplification but also more distortion. Default is configurable, but is typically `0.3`. Cannot be greater than `10.0`. To disable this optimization, use a value of `0`.
- `watermarks` (Array of Objects) Objects describing watermarks which should be applied to page content. Each item must be an object which conforms to the following:
 - *text watermark:*
 - `type`: `"text"` (String) **Required.** Must be set to `"text"` to indicate the object represents a text watermark.
 - `text` (String) Actual text of the watermark. Within the string, you can use the following special tokens to insert dynamic values:
 - `{{pageNumber}}` - Will be replaced with the current page number.
 - `{{pageCount}}` - Will be replaced with the total number of pages.
 - `opacity` (Number) Opacity of the watermark. `1.0` is completely opaque, `0.0` is completely transparent. Default is `1.0`.
 - `color` (String) Text color. Can be any valid CSS color name (like `"red"`) or hex value (like `"#FF0000"`). Default is `"black"`.
 - `fontFamily` (String) Font family for the text. Default for SVG output is to use the browser's default font. Default for raster output is unspecified.
 - `fontSize` (String) Font size specified in points (like `"12pt"`). Default for SVG output is to use the browser's default font size. Default for raster output is unspecified.
 - `fontWeight` (String) Determines the font weight. Possible values:
 - `"normal"` (default)
 - `"bold"`
 - `textDecoration` (String) Possible values:
 - `"none"` (default)
 - `"underlined"`
 - `horizontalAlign` (String) Determines the horizontal position of the watermark. Default is `"center"`. Possible values:
 - `"left"` - Text will be horizontally anchored to the left side of the page and text will be left aligned.
 - `"center"` - Text will be horizontally anchored to the center of the page and text will be centered. (default)
 - `"right"` - Text will be horizontally anchored to the right side of the page and text will be right aligned.
 - `verticalAlign` (String) Determines the vertical position of the watermark. Default is `"middle"`. Possible values:
 - `"top"` - Text will be vertically anchored to the top of the page.
 - `"middle"` - Text will be vertically anchored to the middle of the page. (default)
 - `"bottom"` - Text will be vertically anchored to the bottom of the page.
 - *diagonal text watermark:*



- `type: "diagonalText"` (String) **Required.** Must be set to `"diagonalText"` to indicate the object represents a diagonal text watermark.
- `text` (String) Actual text of the watermark. Within the string, you can use the following special tokens to insert dynamic values:
 - `{{pageNumber}}` - Will be replaced with the current page number.
 - `{{pageCount}}` - Will be replaced with the total number of pages.
- `opacity` (Number) Opacity of the watermark. `1.0` is completely opaque, `0.0` is completely transparent. Default is `1.0`.
- `color` (String) Text color. Can be any valid CSS color name (like `"red"`) or hex value (like `"#FF0000"`). Default is `"black"`.
- `fontFamily` (String) Font family for the text. Default for SVG output is to use the browser's default font. Default for raster output is unspecified.
- `fontSize` (String) Font size specified in points (like `"12pt"`). Default for SVG output is to use the browser's default font size. Default for raster output is unspecified.
- `fontWeight` (String) Determines the font weight. Possible values:
 - `"normal"` (default)
 - `"bold"`
- `textDecoration` (String) Possible values:
 - `"none"` (default)
 - `"underlined"`
- `slope` (String) Controls the text angle. Default is `"up"`. Possible values:
 - `"up"` - Text will start in the lower-left corner of the page and extend upwards to the upper-right corner of the page. (default)
 - `"down"` - Text will start in the upper-left corner of the page and extend downwards to the lower-right corner of the page.
- *image watermark:*
 - `type: "image"` (String) **Required.** Must be set to `"image"` to indicate the object represents an image watermark.
 - `opacity` (Number) Opacity of the watermark. `1.0` is completely opaque, `0.0` is completely transparent. Default is `1.0`.
 - `src` (String) **Required.** URL or [work file](#) id of a PNG image to use for this watermark. When using a URL, the URL must be accessible from the server where PrizmDoc Server is running. > **NOTE:** The `src` MUST be a PNG. If you use a different image format, invalid watermarks will be created.
 - `horizontalAlign` (String) Determines the horizontal position of the watermark. Default is `"center"`. Possible values:
 - `"left"` - Image will be horizontally anchored near the left side of the page.
 - `"center"` - Image will be horizontally anchored to the center of the page. (default)
 - `"right"` - Image will be horizontally anchored near the right side of the page.
 - `verticalAlign` (String) Determines the vertical position of the watermark. Default is `"middle"`. Possible values:
 - `"top"` - Image will be vertically anchored near the top of the page.
 - `"middle"` - Image will be vertically anchored to the middle of the page. (default)
 - `"bottom"` - Image will be vertically anchored near the bottom of the page.
 - `scale` (Number) - Determines the relative size of the image as compared to the size of the page. Value must be between `0.0` and `1.0`. A value of `1.0` indicates the image will be scaled to the size of the page while `0.0` indicates the image will be scaled infinitesimally small and will not be rendered. Default is `0.25`.
 - `autoSize` (String) When set, the image



will be automatically sized to fill the page (any value provided for `scale`, `horizontalAlign`, and `verticalAlign` will be ignored). Possible values:



- `"fit"` - Image will be scaled to be as large as possible while still completely fitting within the page. The aspect ratio of the image is maintained.
 - `"fill"` - Image will be scaled to be large enough that the entire page is covered by the image. Some of the image may fall off the edge of the page, but the entire page is guaranteed to be covered by some part of the image. The aspect ratio of the image is maintained.
 - `"stretch"` - Image width and height will be independently resized so that the image width and height are the same as the page. The aspect ratio of the image is ignored.
- `pageContentEncryption` (String) - Controls whether or not page content will be encrypted for the viewing session. See the [Enabling Content Encryption](#) topic for more information about this feature. Possible values:
 - `"default"` - Product configuration will be used to determine whether or not page content will be encrypted (see `viewing.contentEncryption.enabled` in the [Central Configuration](#) file).
 - `"enabled"` - Page content will be encrypted for the viewing session.
 - `"disabled"` - Page content will not be encrypted for the viewing session.
- `countOfInitialPages` (Integer) Number of pages which should be eagerly converted, or 0 if all pages should be eagerly converted. Default is 0.
- `startConverting` (String) When the `documentSource` is `"http"` or `"file"`, controls whether initial pages should be converted as soon as the document has been acquired. Default is `"none"`. Possible values:
 - `"none"` - Conversion will begin only after the session is explicitly started or page content or attributes are requested. (default)
 - `"initialPages"` - Conversion will begin as soon as the source document has been acquired.
- `contentType` (String) - Determines what kind of browser content will be eagerly pre-generated (other kinds of content may still be generated if explicitly requested). Possible values:
 - `"svgb"` - Pre-generate fully-optimized SVG (uses a unicode inline font to store glyph definitions). Smallest possible SVG, but may not be compatible with some browsers. Recommended whenever possible.
 - `"svga"` - Pre-generate partially-optimized SVG (uses a non-unicode inline font to store only the most frequently-occurring glyph definitions). May not be compatible with some browsers. Use only if `"svgb"` content is not compatible with the target browser.
 - `"svg"` - Pre-generate unoptimized SVG (no font is used; glyph definitions are expressed as SVG path operations). Broadest compatibility with browsers but typically *much* larger, so it renders and scrolls much slower than `"svgb"` and `"svga"`. Not recommended. Use only as a fallback if both `svgb` and `svga` are not compatible with the target browser, or the use of webfonts is disabled in the target browser.
 - `"png"` - Pre-generate raster content.
- `serverCaching` (String) Controls whether output is kept for potential reuse by other viewing sessions. Default is `"full"`. Possible values:
 - `"full"` - Output will be written to disk on the server and retained for reuse by other viewing sessions created for the same source document. Output will not be deleted until the configured viewing cache lifetime is reached (which is a full day with an out-of-box configuration; see `viewing.cacheLifetime` in [Central Configuration](#)). Saves processing time if a source document is viewed repeatedly before the cached data is deleted, but does consume more disk space. (default)
 - `"none"` - Output will be written to disk on the server but only retained for the duration of the viewing session and never shared with other viewing sessions. Once the viewing session expires, the output will be deleted from the disk. Saves disk space if you know that it is unlikely a source document will ever be viewed more than once, but can result in redundant processing if the same

source document is viewed repeatedly.

- `serverSideSearch` (String) Determines whether the server-side search feature will be available for the viewing session. Default is `"enabled"`. Possible values:
 - `"enabled"` - Server-side search will be available for the viewing session. (default)
 - `"disabled"` - Server-side search will not be available for the viewing session.
- `attachmentIndex` (Integer) - *Intended for use only by PrizmDoc Server when it automatically creates viewing sessions for attachments. This is not a property your application should use.* If the source document is an attachment that belongs to another document (such as an email), the 1-based index of this attachment in the list of all attachments (e.g. 1 means it was the first attachment, 2 means it was the second, etc.) or 0 to indicate that the source document is not an attachment. Default is 0.
- `attachmentDisplayName` (String) - *Intended for use only by PrizmDoc Server when it automatically creates viewing sessions for attachments. This is not a property your application should use.* If the source document is an attachment that belongs to another document (such as an email), the filename of the attachment or `null`. Default is `null`.

Successful Response

Response Body

JSON with metadata about the created viewing session.

- `viewingSessionId` (string) Unique id for this viewing session.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
403	<code>"InvalidSecret"</code>	The provided <code>Accusoft-Secret</code> request header value did not match the value of <code>secretKey</code> in your PAS config file . Only applies when you are self-hosting PAS.
404	<code>"DocumentNotFound"</code>	The specified document could not be found.
480	<code>"InvalidInput"</code>	Input is invalid. See <code>errorDetails</code> in the response body.
480	<code>"CannotChangeDocument"</code>	The viewing session already has an <code>original</code> source document.
480	<code>"IncorrectUsage"</code>	The viewing session is already used for viewing of a single source document.
480	<code>"FeatureNotLicensed"</code>	MSO enabled license is not installed.
480	<code>"FeatureDisabled"</code>	Microsoft Office renderer is not configured.
480	<code>"InputNotSupportedWithDocumentId"</code>	Input is not supported when creating a viewing session which auto-generates an associated viewing package in the background. See <code>errorDetails</code> in the response body.
480	<code>"InputConflictsWithViewingPackage"</code>	Input conflicts with the value previously used to create the associated viewing package. See <code>errorDetails</code> in the response body.

Status Code	JSON <code>errorCode</code>	Description
480	"CouldNotGetRemoteResource"	Could not download a document from the specified source <code>url</code> .
580	"CouldNotConnectToRemoteResource"	Could not establish a connection with the specified source <code>url</code> .

Examples

Uploading the source document in a subsequent request

```
POST pas_base_url/ViewingSession
Content-Type: application/json

{
  "source": {
    "type": "upload",
    "displayName": "sample_2015-10-31T19:15:32Z.doc"
  }
}
```

Using a URL for the source document

```
POST pas_base_url/ViewingSession
Content-Type: application/json

{
  "source": {
    "type": "url",
    "url": "http://myserver/mydocument.docx",
    "headers": {
      "My-Custom-Header": "some value required by my server"
    },
    "acceptBadSslCertificate": false
  }
}
```

Using an existing viewing package

You can use an already-created viewing package to start a viewing session. The pre-converted viewable content will be immediately available and no work will be sent to PrizmDoc Server. See the [PAS Viewing Package Creators API](#) for more details about creating viewing packages and the [PAS Viewing Packages API](#) for more details about managing viewing packages.

```
POST pas_base_url/ViewingSession
Content-Type: application/json

{
  "source": {
    "type": "viewingPackage",
    "documentId": "doc_9495837910qc"
  }
}
```

```
}
```

Dynamically creating a viewing package in the background the first time a document is viewed

It is possible to ask PAS to use a viewing package if it exists and, if not, automatically start creating such a viewing package in the background so that it can be used by future viewing sessions. You do this by simply specifying a `documentId`.

For example, here is a POST which uses a `source.type` of "upload" but also specifies a `documentId`:

```
POST pas_base_url/ViewingSession
Content-Type: application/json
{
  "source": {
    "type": "url",
    "url": "http://myserver/mydocument.docx",
    "headers": {
      "My-Custom-Header": "some value required by my server"
    },
    "documentId": "doc_9495837910qc"
  }
}
```

If a viewing package exists for the given `documentId`, then it will be used and no conversion work will be performed. If a viewing package does *not* exist for the given `documentId`, then PAS will use PrizmDoc Server to prepare the content needed for this viewing session *and* also start creating a viewing package in the background. The application can continue to create viewing sessions in the same way for this document. Once the viewing package exists, PAS will simply use the contents of the package directly and stop sending conversion work to PrizmDoc Server.

Comparing two Microsoft Word documents

```
POST pas_base_url/ViewingSession
Content-Type: application/json
{
  "source": {
    "type": "comparison",
    "original": {
      "type": "upload",
      "displayName": "2019-10-07-original.docx"
    },
    "revised": {
      "type": "upload",
      "displayName": "2019-10-07-revised.docx"
    }
  }
}
```

PUT /ViewingSession/{viewingSessionId}/SourceFile

Routes key: `PutViewingSessionSourceFile`

Uploads a document to the session.

Request**Request Headers**

Name	Description
<code>Accusoft-Secret</code>	Required only if you are self-hosting PAS , must match the value of <code>secretKey</code> in your PAS config file .

Request Body

The bytes of the source document.

Successful Response

A simple HTTP 200 status code indicating the file was received.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
403	"InvalidSecret"	The provided <code>Accusoft-Secret</code> request header value did not match the value of <code>secretKey</code> in your PAS config file . Only applies when you are self-hosting PAS.

Example**Request**

```
PUT pas_base_url/ViewingSession/u{viewingSessionId}/SourceFile
Accusoft-Secret: mysecretkey

<<file bytes>>
```

Response

```
HTTP/1.1 200 OK
```

PUT /v2/viewingSessions/{viewingSessionId}/sourceFile/original**Routes key:** `PutViewingSessionOriginalSourceFile`

Used when viewing a comparison of two documents, uploads the first of the two documents, the *original* document.

Request

Request Headers

Name	Description
Accusoft-Secret	Required only if you are self-hosting PAS , must match the value of <code>secretKey</code> in your PAS config file .

Request Body

The bytes of the *original* document (for comparison with a *revised* document).

Successful Response

A simple HTTP 200 status code indicating the file was received.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No viewing session with the provided <code>viewingSessionId</code> could be found.
403	"InvalidSecret"	The provided <code>Accusoft-Secret</code> request header value did not match the value of <code>secretKey</code> in your PAS config file . Only applies when you are self-hosting PAS.
480	"CannotChangeDocument"	The viewing session already has an <i>original</i> document assigned.
480	"UnsupportedFormatForComparison"	The uploaded file was not a Word document (for comparison viewing, we currently only support "doc" and "docx" files).
480	"IncorrectUsage"	For any new viewing session, you can give it either 1) a single <i>source</i> document for viewing or 2) two documents (<i>original</i> and <i>revised</i>) which should be viewed as a comparison, but you cannot do both. If you receive this error from this URL, it is because a single <i>source</i> document has already been provided.
480	"FeatureNotLicensed"	The server's license does not allow the use of the MSO (Microsoft Office) feature, so document comparison is not possible.
480	"FeatureDisabled"	The server has not been configured to allow the use of the Microsoft Office renderer, so document comparison is not possible.

Example

Request

```
PUT pas_base_url/v2/viewingSessions/{viewingSessionId}/sourceFile/original
Accusoft-Secret: mysecretkey
```

```
<<file bytes>>
```

Response

```
HTTP/1.1 200 OK
```

PUT /v2/viewingSessions/{viewingSessionId}/sourceFile/revise

Routes key: `PutViewingSessionRevisedSourceFile`

Used when viewing a comparison of two documents, uploads the second of the two documents, the *revised* document.

Request

Request Headers

Name	Description
Accusoft-Secret	Required only if you are self-hosting PAS , must match the value of <code>secretKey</code> in your PAS config file .

Request Body

The bytes of the *revised* document (for comparison with an *original* document).

Successful Response

A simple HTTP 200 status code indicating the file was received.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No viewing session with the provided <code>viewingSessionId</code> could be found.
403	"InvalidSecret"	The provided <code>Accusoft-Secret</code> request header value did not match the value of <code>secretKey</code> in your PAS config file . Only applies when you are self-hosting PAS.
480	"CannotChangeDocument"	The viewing session already has a <i>revised</i> source document assigned.
480	"UnsupportedFormatForComparison"	The uploaded file was not a Word document (for comparison viewing, we currently only support "doc" and "docx" files).
480	"IncorrectUsage"	For any new viewing session, you can give it either 1) a single <i>source</i> document for viewing or 2) two documents

Status Code	JSON <code>errorCode</code>	Description
		<i>(original and revised)</i> which should be viewed as a comparison, but you cannot do both. If you receive this error from this URL, it is because a single <i>source</i> document has already been provided.
480	"FeatureNotLicensed"	The server's license does not allow the use of the MSO (Microsoft Office) feature, so document comparison is not possible.
480	"FeatureDisabled"	The server has not been configured to allow the use of the Microsoft Office renderer, so document comparison is not possible.

Example

Request

```
PUT pas_base_url/v2/viewingSessions/{viewingSessionId}/sourceFile/revised
Accusoft-Secret: mysecretkey

<<file bytes>>
```

Response

```
HTTP/1.1 200 OK
```

GET /ViewingSession/u{viewingSessionId}/SourceFile?ContentDispositionFilename={ContentDispositionFilename}

Routes key: `GetSourceFile`

Gets the source document in use for a viewing session.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.
{ContentDispositionFilename}	The filename as a URL-encoded string, without extension, to be used in the <code>Content-Disposition</code> response header (appropriate file extension such as <code>doc</code> or <code>docx</code> will automatically be added). By default, the value will be <code>SourceFile.<ext></code> .

Response Headers

Name	Description
Content-Disposition	Indicates to a browser that the response body should be treated as a file download and specifies the filename the browser should use.
Content-Type	The most-specific MIME type for the returned document or <code>application/octet-stream</code> otherwise.

Response Body

The raw bytes of the viewing session's source document.

Example

Request

```
GET pas_base_url/ViewingSession/uXYZ.../SourceFile?
ContentDispositionFilename=MonthlySalesReport
```

Response

```
200 OK
Content-Type: application/msword
Content-Disposition: attachment; filename=MonthlySalesReport.docx; filename*=UTF-8' 'MonthlySalesReport.docx

<<file bytes>>
```

GET /v2/viewingSessions/{viewingSessionId}/sourceFile/original?contentDispositionFilename={contentDispositionFilename}

Routes key: `GetOriginalSourceFile`

When viewing a comparison of two documents, gets the *original* document used in the comparison. The document returned will be an identical copy of the document originally provided.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.
{contentDispositionFilename}	The filename as a URL-encoded string, without extension, to be used in the <code>Content-Disposition</code> response header (appropriate file extension such as <code>doc</code> or <code>docx</code> will automatically be added). By default, the value will be <code>OriginalSourceFile.<ext></code> .

Response Headers

Name	Description
Content-Disposition	Indicates to a browser that the response body should be treated as a file download and specifies the filename the browser should use.
Content-Type	The most-specific MIME type for the returned document or <code>application/octet-stream</code> otherwise.

Response Body

The raw bytes of the first of the two documents being viewed as a comparison, the *original* document.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No viewing session with the provided <code>viewingSessionId</code> could be found.
480	"DocumentNotProvidedYet"	An <i>original</i> document has not been provided to the viewing session yet.
480	"IncorrectUsage"	For any new viewing session, you can give it either 1) a single <i>source</i> document for viewing or 2) two documents (<i>original</i> and <i>revised</i>) which should be viewed as a comparison, but you cannot do both. If you receive this error from this URL, it is because a single <i>source</i> document was provided for viewing, so there will never be an <i>original</i> comparison document to get.

Example

Request

```
GET pas_base_url/v2/viewingSessions/XYZ.../sourceFile/original?
contentDispositionFilename=OldMonthlySalesReport
```

Response

```
200 OK
Content-Type: application/msword
Content-Disposition: attachment; filename=OldMonthlySalesReport.docx;
filename*=UTF-8''OldMonthlySalesReport.docx

<<file bytes>>
```

**GET /v2/viewingSessions/{viewingSessionId}/sourceFile/revised?
contentDispositionFilename={contentDispositionFilename}**

Routes key: `GetRevisedSourceFile`

When viewing a comparison of two documents, gets the *revised* document used in the comparison. The document returned will be an identical copy of the document originally provided.

The response will set the `Content-Type` header to the most-specific MIME type it can or `application/octet-stream` otherwise.

Request**URL Parameters**

Parameter	Description
<code>{viewingSessionId}</code>	The <code>viewingSessionId</code> which identifies the viewing session.
<code>{contentDispositionFilename}</code>	The filename as a URL-encoded string, without extension, to be used in the <code>Content-Disposition</code> response header (appropriate file extension such as <code>doc</code> or <code>docx</code> will automatically be added). By default, the value will be <code>RevisedSourceFile.<ext></code> .

Response Headers

Name	Description
<code>Content-Disposition</code>	Indicates to a browser that the response body should be treated as a file download and specifies the filename the browser should use.
<code>Content-Type</code>	The most-specific MIME type for the returned document or <code>application/octet-stream</code> otherwise.

Response Body

The raw bytes of the second of the two documents being viewed as a comparison, the *revised* document.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No viewing session with the provided <code>viewingSessionId</code> could be found.
480	<code>"DocumentNotProvidedYet"</code>	A "revised" document has not been associated with the viewing session yet.
480	<code>"IncorrectUsage"</code>	For any new viewing session, you can give it either 1) a single <i>source</i> document for viewing or 2) two documents (<i>original</i> and <i>revised</i>) which should be viewed as a comparison, but you cannot do both. If you receive this error from this URL, it is because a single <i>source</i> document was provided for viewing, so there will never be a <i>revised</i> comparison document to get.

Example

Request

```
GET pas_base_url/v2/viewingSessions/XYZ.../sourceFile/revise?
contentDispositionFilename=NewMonthlySalesReport
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/msword
Content-Disposition: attachment; filename=NewMonthlySalesReport.docx;
filename*=UTF-8''NewMonthlySalesReport.docx

<<file bytes>>
```

GET /v2/viewingSessions/{viewingSessionId}/restrictions

Returns information about any restrictions enforced by the server for the current viewing session.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.

Successful Response

JSON with information about any restrictions currently in place for the viewing session:

- `delayEnabled` (Boolean) - Indicates whether the server has enforced an artificial delay before the document conversion results are allowed to be delivered to the viewer.
- `delaySecondsRemaining` (Integer) - The number of seconds remaining in the artificially-imposed delay before the document conversion results are allowed to be delivered to the viewer. Only present when `delayEnabled` is true.
- `downloadDisabled` (Boolean) - Indicates whether downloading of the source document has been disabled for this viewing session.
- `markupSavingDisabled` (Boolean) - Indicates whether saving markup has been disabled for this viewing session.
- `markupLoadingDisabled` (Boolean) - Indicates whether loading markup has been disabled for this viewing session.
- `contentConvertersDisabled` (Boolean) Indicates whether the content converters feature has been disabled for this viewing session.
- `markupBurnersDisabled` (Boolean) Indicates whether the markup burner feature has been disabled for this viewing session.
- `formExtractorsDisabled` (Boolean) Indicates whether the forms extractors feature has been disabled for this viewing session.
- `formInfoDisabled` (Boolean) Indicates whether the form detection feature has been disabled for this viewing session.

Example Responses

When the product is licensed and the form detection feature is available

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "delayEnabled": false,
  "downloadDisabled": false,
  "markupSavingDisabled": false,
  "markupLoadingDisabled": false,
  "contentConvertersDisabled": false,
  "markupBurnersDisabled": false,
  "formExtractorsDisabled": false,
  "formInfoDisabled": false
}
```

When the product is unlicensed (evaluation mode), and 9 seconds of initial delay are remaining:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "delayEnabled": true,
  "delaySecondsRemaining": 9,
  "downloadDisabled": true,
  "markupSavingDisabled": true,
  "markupLoadingDisabled": true,
  "contentConvertersDisabled": true,
  "markupBurnersDisabled": true,
  "formExtractorsDisabled": true,
  "formInfoDisabled": true
}
```

POST /ViewingSession/u{viewingSessionId}/Replacement

Routes key: CreateViewingSessionReplacement

Replace the existing viewing session with a new one that has a new `password` parameter value. The other original viewing session parameters are preserved. If a source document is uploaded it is attached to the newly created viewing session with a new password. The old viewing session is stopped.

The replacement API is useful when a session is errored because of either an invalid or missing password, and you want to replace it with a new session with the same parameters against the same document but with a modified password.

Request

Request Headers

Name	Description
Content-Type	Should be <code>application/json</code>

Request Body

- `password` (String) Password to use when opening a password-protected source document. If the parameter is not provided a new viewing session is created with a default null `password` parameter value.

URL Parameters

Parameter	Description
<code>{viewingSessionId}</code>	The <code>viewingSessionId</code> which identifies the viewing session.

Successful Response

Response Body

JSON with metadata about the created viewing session.

- `viewingSessionId` (String) Unique id for this viewing session.

Error Responses

Status Code	Reason Phrase	Description
403	The session is invalid or has expired	You requested a valid <code>{viewingSessionId}</code> but it is no longer available.
500	Internal Server Error	Can occur if you forget to prefix the <code>{viewingSessionId}</code> portion of the URL with <code>u</code> , or if you simply request an invalid <code>{viewingSessionId}</code> .

Some error responses will have a JSON body with an `errorCode` and `errorDetails`:

Status Code	JSON <code>errorCode</code>	Description
480	"PropertyNotReplaceable"	Unsupported property to replace was used. See <code>errorDetails</code> in the response body.

Example

Request

```
POST pas_base_url/ViewingSession/uXYZ.../Replacement
Content-Type: application/json

{
  "password": "123"
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8

{
  "viewingSessionId": "THMMytF4ACHrxmN2bXj4vahx4Gnwly_kEeFt20XtRau-
z43pPHIjUy5JXJ05Wj1MaqvdfsXp98JxIk7ALWkukg"
}
```

Viewing Package Creators

Introduction

The *viewing package creators* REST API allows your application to create new viewing packages (to manage existing viewing packages, use the [viewing packages](#) REST API).

A *viewing package* contains fully pre-converted, browser-ready content for a document. If you know ahead of time which documents are going to be viewed and you want to make the viewing experience as fast as possible for your users, you can create *viewing packages* ahead of time for your documents. Later, when you need to create a *viewing session* for a document, simply use the same `documentId` you used when creating the viewing package and PAS will use the pre-converted content from the viewing package.

NOTE: Creating viewing packages requires you to configure a database. See [PAS Configuration](#) for more information.

Available URLs

URL	Description
POST /v2/viewingPackageCreators	Creates a new viewing package creator process.
PUT /v2/viewingPackageCreators/{processId}/SourceFile	Uploads the source document to use when creating the viewing package.
GET /v2/viewingPackageCreators/{processId}	Gets the status of a viewing package creator process.

POST /v2/viewingPackageCreators

Starts a new viewing package creator process.

Request

Request Headers

Name	Description
<code>Accusoft-Secret</code>	Required only if you are self-hosting PAS , must match the value of <code>secretKey</code> in your PAS config file .

Request Body

JSON object conforming to the following:

Property	Type	Required	Description
<code>input.source</code>	{object}	Yes	Properties relevant to PAS will reside in this object. > NOTE: This object will be removed before sending the rest of the body to the PrizmDoc Services.
<code>input.source.documentId</code>	{string}	Yes	A customer supplied identifier which is used to uniquely identify the resulting viewing package. The documentId is required to have a non-zero length less than 256 characters and must consist of characters defined by RFC4648 - Base 64 Encoding with URL and Filename Safe Alphabet .
<code>input.source.type</code>	{string}	Yes	Specify where PrizmApplicationServices can find the document. The valid values are "document", "url", or "upload".
<code>input.source.fileName</code>	{string}	Yes (If source.type="document")	Filename that the Storage Provider API can use.
<code>input.source.url</code>	{string}	Yes (If source.type="url")	URL to create a viewing session from.
<code>input.source.headers</code>	{object}	Optional (Used if source.type="url")	Request headers to send from PrizmApplicationServices when retrieving the URL.
<code>input.source.acceptBadSslCertificate</code>	{boolean=false}	Optional (Used if source.type="url")	If true, requires SSL certificates be valid. > NOTE: to use your own certificate authority (CA), you need to specify an agent that was created with that CA as an option.

Property	Type	Required	Description
<code>input.source.displayName</code>	{string}	Yes (if <code>source.type="upload"</code>)	The display name of the document that's being uploaded. E.g. "sample.doc"
<code>input.source.markupId</code>	{string}	Optional	The ID to use when querying markup with the XML and JSON layer APIs. If the ID is not provided, we create a new one from the other information we are given. The <code>markupId</code> is required to have a non-zero length, should be less than 256 characters, and should consist of characters defined by RFC4648 - Base 64 Encoding with URL and Filename Safe Alphabet .
<code>input.source.downloadName</code>	{string}	Optional	The name that will be used when downloading the original document. We will use the "displayName" or "fileName" if this option is not provided.
<code>input.source.fileExtension</code>	{string}	Optional	A file extension that is used if the File Detection Service cannot determine what type of file was uploaded. We will get the extension from "displayName" or "fileName" if this option is not provided. > NOTE: <i>This parameter may only include alpha-numeric characters.</i>
<code>input.viewingPackageLifetime</code>	{integer}	Optional	Defines the minimum number of integer seconds for the created content to remain available. By default this is 24 hours (if not configured differently through PAS Configuration). If set to 0, the content will remain available

Property	Type	Required	Description
			perpetually.
<code>input.render</code>	{object}	Optional	The object that describes rendering options.
<code>input.render.html5</code>	{object}	Optional	The object that describes rendering options for the HTML5 viewer. A whitelist of the following render properties is used: <code>svgMaxImageSize</code> and <code>vectorTolerance</code> . They are described in the PrizmDoc Server Viewing Session documentation.
<code>minSecondsAvailable</code>	{integer}	Optional	The minimum number of seconds this viewing package creator will remain available so you can GET its status. The actual lifetime may be longer.

Examples

Create a Viewing Package using a local document

```
POST pas_base_url/v2/viewingPackageCreators
Content-Type: application/json

{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.doc",
      "documentId": "unT67FxeKm81klp0kPnyg8",
      ...
    },
    "viewingPackageLifetime": 2592000
  }
}
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "source": {
      "type": "document",
```

```
        "fileName": "sample.doc",
        "documentId": "unT67FxeKm81k1p0kPnyg8",
        ...
    },
    "viewingPackageLifetime": 2592000
},
"expirationDateTime": "2015-12-09T06:22:18.624Z",
"processId": "khjyrfKLj2g6gv8fdqg710",
"state": "processing",
"percentComplete": 0
}
```

The response, along with other information, includes a new processId which is used to GET information about the converted content.

Error Responses

Duplicate POST Prior to Process Completion:

If a POST request is made with a body containing the exact same content as a request prior to the completion of the original process, an error similar to the one below will be returned.

```
HTTP/1.1 580 ConversionInProgress
Content-Type: application/json

{
  "errorCode": "ConversionInProgress"
}
```

Duplicate POST After Process Completion:

If a POST request is made with a body containing the exact same content as a prior request after that request has completed, the following error similar to the one below will be returned.

```
HTTP/1.1 580 DocumentIdAlreadyInUse
Content-Type: application/json

{
  "errorCode": "DocumentIdAlreadyInUse"
}
```

NOTE: An existing package must be deleted before re-submitting the same content for conversion.

Examples of Input Validation error responses:

```
HTTP/1.1 480 MissingInput
Content-Type: application/json

{
  "errorCode": "MissingInput",
  "errorDetails": {
    "in": "body",
    "at": "input"
  }
}
```

```
HTTP/1.1 480 InvalidInput
Content-Type: application/json
```

```
{
  "errorCode": "InvalidInput",
  "errorDetails": {
    "in": "body",
    "at": "input",
    "expected": {
      "type": "object"
    }
  }
}
```

If you are self-hosting PAS and a POST request does not include the correct `Accusoft-Secret` header value, you will receive an error like the one below. See [PAS Configuration](#) for more info about configuring your PAS `secretKey`.

```
HTTP/1.1 403 InvalidSecret
Content-Type: application/json
```

```
{
  "errorCode": "InvalidSecret"
}
```

Create a Viewing Package with extra options

PrizmDoc Server allows for render options that modify the output of the conversion process. The valid options are documented at the top of this page, but as an example, this is what a request would look like if you only wanted raster content:

NOTE: When creating a viewing package, watermarks cannot be used and will return an error.

```
POST pas_base_url/v2/viewingPackageCreators
Content-Type: application/json
```

```
{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.doc",
      "documentId": "unT67FxeKm8lk1p0kPnyg8",
      ...
    },
    "viewingPackageLifetime": 2592000
  }
}
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
```

```

    "input": {
      "source": {
        "type": "document",
        "fileName": "sample.doc",
        "documentId": "unT67FxeKm81k1p0kPnyg8",
        ...
      },
      "viewingPackageLifetime": 2592000
    },
    "expirationDateTime": "2015-12-09T06:22:18.624Z",
    "processId": "khjyrfKLj2g6gv8fdqg710",
    "state": "processing",
    "percentComplete": 0
  }

```

Error Responses

Watermarks Input:

The `input.watermarks` property cannot be used when creating a viewing package.

```

HTTP/1.1 480 ReservedInput
Content-Type: application/json

{
  "errorCode": "ReservedInput",
  "errorDetails": {
    "in": "body",
    "at": "input.watermarks"
  }
}

```

PUT /v2/viewingPackageCreators/{processId}/SourceFile

Uploads the source document to use when creating the viewing package.

Request

Request Headers

Name	Description
Accusoft-Secret	Required only if you are self-hosting PAS , must match the value of <code>secretKey</code> in your PAS config file .

Request Body

The source document to use.

A request to this URL allows the user to upload a document to an existing viewing package creator which has not yet been provided a source document. When a viewing package creator is configured with `input.source.type: 'upload'`, viewing package creation waits until a PUT request containing the source document is received. After the source document has been successfully uploaded, viewing package creation will resume.

Examples

```
PUT pas_base_url/v2/viewingPackageCreators/khjyrfKLj2g6gv8fdqg710/SourceFile
Content-Type: application/octet-stream

<<file bytes>>
```

Successful Response

```
HTTP/1.1 200 OK
```

Error Responses

When uploading a document for viewing package creation, the following errors may be encountered:

PUT When Not Allowed

If a PUT request is made to upload a file for a viewing package creator which does not have `input.source.type: 'upload'`, then the following error will be returned.

```
HTTP/1.1 580 NotSupportedForViewingPackageSourceType
Content-Type: application/json

{
  "errorCode": "NotSupportedForViewingPackageSourceType"
}
```

PUT When File Already Provided

If a PUT request is made to upload a file for a viewing package creator after a file has already been uploaded for that viewing package creator, then the following error will be returned.

```
HTTP/1.1 580 SourceFileAlreadyProvided
Content-Type: application/json

{
  "errorCode": "SourceFileAlreadyProvided"
}
```

If you are self-hosting PAS and a PUT request does not include the correct `Accusoft-Secret` header value, you will receive an error like the one below. See [PAS Configuration](#) for more info about configuring your PAS `secretKey`.

```
HTTP/1.1 403 InvalidSecret
Content-Type: application/json

{
  "errorCode": "InvalidSecret"
}
```

GET /v2/viewingPackageCreators/{processId}

Gets the status of a viewing package creator process.

Requests to this URL can be sent repeatedly while the state="processing". To limit network congestion, an exponential back-off algorithm is recommended. This means that the time interval between each request is increased, which results in a good trade-off between quickly discovering short conversions that have completed and preventing a large number of requests for long conversions.

Upon successful package creation, the response will have the state="complete" with percentComplete=100. The errorCode will be null. Additionally it will include an output property which will contain the created package expiration time. If there is an error during processing, the state="error" with percentComplete=100. The exception to this is, some but not all page failures. In this case the state is set to "complete".

Request

Request Headers

Name	Description
Accusoft-Secret	Required only if you are self-hosting PAS , must match the value of <code>secretKey</code> in your PAS config file .

Examples

```
GET pas_base_url/v2/viewingPackageCreators/khjyrfKLj2g6gv8fdqg710
```

Viewing package creation completed successfully

A response for a package that was successfully generated would look like this:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.doc",
      "documentId": "unT67FxeKm8lk1p0kPnyg8",
      . . .
    },
    "viewingPackageLifetime": 2592000
  },
  "output": {
    "packageExpirationDateTime": "2016-1-09T06:22:18.624Z"
  },
  "expirationDateTime": "2015-12-09T06:22:18.624Z",
  "processId": "khjyrfKLj2g6gv8fdqg710",
  "state": "complete",
  "percentComplete": 100
}
```

The viewing package creation process may encounter three types of errors, which it will report on as follows:

Source document not found

This error describes the conversion result for a nonexistent document. Note that upon completion in this error case, the state is set to "error" because no content could be provided for the document. Additionally, an `errorCode` property will be provided indicating this is the cause of the error. No output property will be provided when the process completes in an error state.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.doc",
      "documentId": "unT67FxeKm81k1p0kPnyg8",
      . . .
    },
    "viewingPackageLifetime": 2592000
  },
  "expirationDateTime": "2015-12-09T06:22:18.624Z",
  "processId": "khjyrfKLj2g6gv8fdqg710",
  "state": "error",
  "percentComplete": 100,
  "errorCode": "DocumentNotFound"
}
```

Incorrect password for source document

This error describes the conversion result for a password-protected document, which is not supported in this release. A password property may be provided in the request, but it will not be used. Upon completion in this error case, the state is set to "error" because no content could be provided for the document. An `errorCode` property will be provided indicating an "InternalError". No output property will be provided when the process completes in an error state.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.pdf",
      "documentId": "unT67FxeKm81k1p0kPnyg8",
      . . .
    },
    "password": "opensesame",
    "viewingPackageLifetime": 2592000
  },
  "expirationDateTime": "2015-12-09T06:22:18.624Z",
  "processId": "khjyrfKLj2g6gv8fdqg710",
  "state": "error",
  "percentComplete": 100,
  "errorCode": "InternalError"
}
```

Individual page conversion failures

These errors will not mark the entire package as errored, but rather report in the `output` object in a `pageFailures`

array.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.doc",
      "documentId": "unT67FxeKm8lk1p0kPnyg8",
      . . .
    },
    "viewingPackageLifetime": 2592000
  },
  "output": {
    "packageExpirationDateTime": "2016-1-09T06:22:18.624Z",
    "pageFailures": [
      {
        "errorCode": "CouldNotRetrievePngContent",
        "pageNumber": "4"
      },
      {
        "errorCode": "CouldNotRetrieveSvgContent",
        "pageNumber": "5"
      },
      {
        "errorCode": "CouldNotRetrieveTextContent",
        "pageNumber": "8"
      }
    ]
  },
  "expirationDateTime": "2015-12-09T06:22:18.624Z",
  "processId": "khjyrfKLj2g6gv8fdqg710",
  "state": "complete",
  "percentComplete": 100
}
```

Attachment failures

These errors describe the failure of the conversion to provide an artifact for an attachment in the case where a document supports attachments, such as .EML or .MSG.

If the conversion process finds attachments associated with the document, but the process is unable to provide content for one or more pages, then the state is set to "complete" and an array of `attachments` is returned as a property in the body.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.eml",
      "documentId": "unT67FxeKm8lk1p0kPnyg8",
      . . .
    },
    "viewingPackageLifetime": 2592000
  },
}
```

```
"output": {
  "packageExpirationDateTime": "2016-1-09T06:22:18.624Z",
  "attachments": [
    {
      "name": "text-file",
      "errorCode": "ArtifactCouldNotBeRetrieved"
    },
    {
      "name": "document-text.pdf"
    },
    {
      "name": "last-pages-dont-exist.pdf",
      "errorCode": "CouldNotConvertSourceDocument"
    }
  ]
},
"expirationDateTime": "2015-12-09T06:22:18.624Z",
"processId": "khjyrfKLj2g6gv8fdqg710",
"state": "complete",
"percentComplete": 100
}
```

Error Responses

If you are self-hosting PAS and a GET request does not include the correct `Accusoft-Secret` header value, you will receive an error like the one below. See [PAS Configuration](#) for more info about configuring your PAS `secretKey`.

```
HTTP/1.1 403 InvalidSecret
Content-Type: application/json

{
  "errorCode": "InvalidSecret"
}
```

Viewing Packages

Introduction

The *viewing packages* REST API allows your application to manage existing viewing packages.

To create a new viewing package, use the [viewing package creators](#) REST API or provide a `documentId` when creating a [viewing session](#).

Available URLs

URL	Description
GET /v2/viewingPackages/{documentId}	Gets info about a viewing package.
GET /v2/viewingPackages/{documentId}/creator	Gets the <code>processId</code> of the <code>viewingPackageCreator</code> which created this viewing package if it still exists.
DELETE /v2/viewingPackages/{documentId}	Deletes a viewing package and its associated content stored on disk.

GET /v2/viewingPackages/{documentId}

Gets info about a viewing package.

Request

Request Headers

Name	Description
Accusoft-Secret	Required only if you are self-hosting PAS , must match the value of <code>secretKey</code> in your PAS config file .

After a viewing package creator process has been created, information about the converted content can be requested as shown in the example below:

Example

```
GET pas_base_url/v2/viewingPackages/unT67FxeKm8lk1p0kPnyg8
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "source": {
    "type": "document",
    "fileName": "sample.doc",
    "documentId": "unT67FxeKm8lk1p0kPnyg8"
  },
  "state": "complete",
  "packageExpirationDateTime": "2016-1-09T06:22:18.624Z",
  "pageFailures": [
    {
      "errorCode": "CouldNotRetrieveJpegContent",
      "pageNumber": "4"
    },
    {
      "errorCode": "CouldNotRetrieveSvgContent",
      "pageNumber": "5"
    }
  ]
}
```

Error Responses

If you are self-hosting PAS and a GET request does not include the correct `Accusoft-Secret` header value, you will receive an error like the one below. See [PAS Configuration](#) for more info about configuring your PAS `secretKey`.

```
HTTP/1.1 403 InvalidSecret
Content-Type: application/json

{
  "errorCode": "InvalidSecret"
}
```

GET /v2/viewingPackages/{documentId}/creator

Gets the `processId` of the `viewingPackageCreator` which created this viewing package if it still exists.

This URL is useful when you [create a viewing session](#) with a `documentId` (allowing PAS to automatically start a `viewingPackageCreator` for you in the background if no viewing package exists yet) and you want to check the status of the `viewingPackageCreator` which PAS may have created on your behalf. After creating a viewing session with a `documentId`, you can use this URL to see if any associated `viewingPackageCreator` exists. If it does, you can then use the returned `viewingPackageCreatorId` to GET more info about the status of the `viewingPackageCreator` that PAS started for you.

Request Headers

Name	Description
Accusoft-Secret	Required only if you are self-hosting PAS , must match the value of <code>secretKey</code> in your PAS config file.

Example

```
GET pas_base_url/v2/viewingPackages/unT67FxeKm81k1p0kPnyg8/creator
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "viewingPackageCreatorId": "khjyrfKLj2g6gv8fdqg710"
}
```

Error Responses

If a request is made with an invalid viewing package ID or the creator process has expired, an error will be returned.

```
HTTP/1.1 404 Not Found
```

If you are self-hosting PAS and a GET request does not include the correct `Accusoft-Secret` header value, you will receive an error like the one below. See [PAS Configuration](#) for more info about configuring your PAS `secretKey`.

```
HTTP/1.1 403 InvalidSecret
Content-Type: application/json

{
  "errorCode": "InvalidSecret"
}
```

DELETE /v2/viewingPackages/{documentId}

Deletes a viewing package and its associated content stored on disk.

Request

Request Headers

Name	Description
Accusoft-Secret	Required only if you are self-hosting PAS , must match the value of <code>secretKey</code> in your PAS config file.

Example

```
DELETE pas_base_url/v2/viewingPackages/unT67FxeKm8lk1p0kPnyg8
```

Successful Response

If a DELETE request is made the content will be marked for asynchronous deletion.

```
HTTP/1.1 204 No Content
```

Error Responses

If a DELETE request is made with an invalid package ID, an error will be returned.

```
HTTP/1.1 404 Not Found
```

If you are self-hosting PAS and a DELETE request does not include the correct `Accusoft-Secret` header value, you will receive an error like the one below. See [PAS Configuration](#) for more info about configuring your PAS `secretKey`.

```
HTTP/1.1 403 InvalidSecret
Content-Type: application/json

{
  "errorCode": "InvalidSecret"
}
```

```
}
```

Self-Hosted Administration

This REST API is useful if you are self-hosting PAS instances:

- [Health](#)

Health

Introduction

For customers who are self-hosting PAS, the *health* REST API allows an administrator or application to check the health of a PAS instance.

NOTE: These URLs are not available in [PrizmDoc Cloud](#).

Available URLs

URL	Description
GET /health	Determines whether PAS is healthy or not.
GET /servicesConnection	Returns the status of PAS connectivity to PrizmDoc Server.
GET /info	A request to get information about the service.

GET /health

Determines whether PAS is healthy or not. A `200` response indicates PAS is healthy. Anything else indicates PAS is unhealthy.

```
GET http://localhost:3000/health
```

Successful Response

```
HTTP/1.1 200 OK  
OK
```

There are no error states for this request. If the request times out or the connection is refused, then the service is not running or reachable.

GET /servicesConnection

Returns the status of PAS connectivity to PrizmDoc Server.

Successful Response

```
HTTP/1.1 200 OK
OK
```

A `200` response indicates PAS is able to communicate with PrizmDoc Server.

NOTE: A successful response only indicates whether PAS is able to communicate with PrizmDoc Server; it does not mean that PrizmDoc Server instances themselves are healthy. To check the health of PrizmDoc Server instances, use the [PrizmDoc Server Health REST API](#).

Error Responses

```
HTTP/1.1 580 Server Error
```

A non-`200` response indicates PAS is NOT able to communicate with PrizmDoc Server.

GET /info

A request to get information about the service.

```
GET http://localhost:3000/info
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "version": "X.X.XXXX.XXXX"
}
```

Where the version is the PAS version.

There are no error states for this request. If the request times out or the connection is refused, then the service is not running or reachable.

Viewer Support

These REST APIs are used by our viewer. It is uncommon for your application to need to use them:

- [Attachments](#)
- [Content Converters](#)
 - [Content Converters \(Deprecated\)](#)
- [Form Definitions](#)
- [Form Extractors](#)

- [Image Stamps](#)
- [Legacy Create Session](#)
- [Markup Burners](#)
- [Markup Layers](#)
- [Markup XML](#)
- [Search Tasks](#)

Attachments

Introduction

The *attachments* REST API is used by our viewer to get EML and MSG attachments for a document being viewed.

Available URLs

URL	Description
GET /ViewingSession/u{ViewingSessionID}/Attachments	Lists the attachments available in the source document.

GET /ViewingSession/u{ViewingSessionID}/Attachments

Routes key: `GetAttachments`

This API returns a JSON list of attachment objects. If the return object list length has a zero count and no errors detected, there are no attachments. The status property of the list object indicates whether or not the list has been completed. If the list hasn't been completed, the request must be retried again. Each Attachment object in the list contains **displayName** and **viewingSessionId** property. The **displayName** can be used to label an href link on a web page and the link points to the **viewingSessionId** URL for the attachment.

Request

URL Parameters

Parameter	Description
{ViewingSessionID}	The ID of the viewing session associated with the request.

Successful Response

Response Body

If successful, this method returns the following properties:

- `status` (String) Specifies the current status of the attachments.
 - "pending" - There may be attachments, but the list has not yet been constructed.
 - "complete" - The list is known and present in this object.
- `attachments` (List) The list of attachments, if any. Each item in this list will be a new viewing session ID, which is used to view any of the listed attachments in the Viewer by passing in the viewing session ID provided in this list.

Error Responses

Status Code	Description
500	Internal service error. This error can be returned for a number of different reasons. Please verify that your input is correct, and contact Support if the error persists.

Examples

Request

```
GET pas\_base\_url/ViewingSession/uGcIsIsEGbLV2_V9yy4NzmK2HB-JuLOH--A9sZ16c1a9tx00ZDBGfq1G4kKu0r_GyEps4wWCvDwn4dpnZAR76Uw/Attachments
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "attachments": [
    {
      "displayName": "example-file.pdf",
      "viewingSessionId":
"SC0fZEMYiiGdOPMKBqLY8P6EAnVLEqxeTHjUUYqqxgJjc3s3wsQ8Lw2qqvkD1uLKTpAlF1ce23EQ6BFpYb4E3LC9TsXxwHCgB-
I1c5rPOt0"
    },
    {
      "displayName": "second-example.doc",
      "viewingSessionId": "33qQovKTjc0UKbNgOI-5POEyCpNw5x-
uEzGMB13iUVhnCa_UHSSnOpRBEzPKed7Maxq2RQu2SOowJj14X4iU_65OQjx2EI5-7h-bX1Yc6uA"
    }
  ],
  "status": "complete"
}
```

Content Converters

Introduction

The *content converters* REST API is used by our viewer to allow the user to download the document they are viewing as a PDF.

This REST API is designed for our viewer. If your application needs to convert a document to a different format, **we recommend you use [PrizmDoc Server's content converters REST API](#) instead**. It offers more options and does not require the use of a viewing session.

Available URLs

URL	Description
POST /v2/viewingSessions/{viewingSessionId}/contentConverters	Starts a conversion process for the source document of a particular viewing session.
GET /v2/viewingSessions/{viewingSessionId}/contentConverters/{processId}	Gets the status of the specific converter process.
GET /v2/viewingSessions/{viewingSessionId}/contentConverters/{processId}/results/{index}/file?ContentDispositionFilename={fileName}	Gets the contents of a file produced by the converter process.

POST /v2/viewingSessions/{viewingSessionId}/contentConverters

Routes key: `CreateContentConverterV2`

Starts a conversion process for the source document of a particular viewing session. This URL has the following parameters:

Parameter	Description
{viewingSessionId}	The ID provided in the response from POST /ViewingSession .

Request

Body: Empty. An error will be returned if any data is present.

NOTE: PDF is currently the only supported destination format. A successful response will include several input properties as shown below. A future release will allow these properties to be set in the request body to create various kinds of output.

Response

Body: a JSON object with the following properties:

Property	Type	Description
input	{object}	An object that specifies the input used for the conversion.
input.dest	{object}	An object that specifies the destination file format and any additional details which control how the content is converted.
input.dest.format	{string}	Specifies the output file format.

Property	Type	Description
<code>input.dest.pdfOptions</code>	{object}	An object that specifies additional options when <code>input.dest.format</code> is "pdf".
<code>input.dest.pdfOptions.forceOneFilePerPage</code>	{boolean}	If true, the conversion process will produce single-page PDF files, one file for each page of content (instead of a single PDF with multiple pages). Default is false.
<code>input.sources</code>	{object}	An array of objects, one for each input file.
<code>input.sources[n].pages</code>	{string}	The page numbers and/or page ranges, separated by commas, of the source document to convert.
<code>processId</code>	{string}	The id of the contentConverter resource which represents the file conversion operation.
<code>state</code>	{string}	The current state of the conversion process, which will be one of the following: "processing", "complete", or "error". If "processing", the conversion is still in progress. If "complete", the conversion has completed successfully. If "error", the conversion failed due to a problem. For the initial POST, this value will almost always be "processing". Results are typically only available with a subsequent GET.
<code>percentComplete</code>	{integer}	An integer from 0 to 100 that indicates what percentage of the conversion is complete.
<code>errorCode</code>	{string}	An error code string if a problem occurred during the conversion process.

Examples

```
POST pas\_base\_url/v2/viewingSessions/ZGZhc2RmYXNkZmFzZGZkcw/contentConverters
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    },
    "sources": [
      {
        "pages": ""
      }
    ]
  },
  "expirationDateTime": "2015-11-04T19:20:09.280Z",
  "processId": "mxivIVSw7UhtLlyWwt3QEA",
  "state": "processing",
  "percentComplete": 0
}
```

Errored Responses

When a response with a status code of 580 is received check the response body for details:

```
HTTP/1.1 580 InternalError
Content-Type: application/json

{
  "errorCode": "InternalError"
}
```

When a viewing session does not exist for the given `viewingSessionId`:

```
HTTP/1.1 404 Not Found
```

When any data is present in the body of the request:

```
HTTP/1.1 480 ReservedInput
Content-Type: application/json

{
  "errorCode": "ReservedInput",
  "errorDetails": {
    "in": "body"
  }
}
```

When the server's license could not be verified:

```
HTTP/1.1 480 LicenseCouldNotBeVerified
Content-Type: application/json

{
  "errorCode": "LicenseCouldNotBeVerified"
}
```

GET /v2/viewingSessions/{viewingSessionId}/contentConverters/{processId}**Routes key:** PollContentConverterV2

Gets the status of the specific converter process. This URL has the following parameters:

Parameter	Description
{viewingSessionId}	The ID provided in the response from POST /ViewingSession.
{processId}	The ID provided in the response from POST /v2/viewingSessions/{viewingSessionId}/contentConverters.

Response**Body:** a JSON object with the following properties:

Property	Type	Description
input	{object}	An object that specifies the input used for the conversion.
input.dest	{object}	An object that specifies the destination file format and any additional details which control how the content is converted.
input.dest.format	{string}	Specifies the output file format.
input.dest.pdfOptions	{object}	An object that specifies additional options when <code>input.dest.format</code> is "pdf".
input.dest.pdfOptions.forceOneFilePerPage	{boolean}	If true, the conversion process will produce single-page PDF files, one file for each page of content (instead of a single PDF with multiple pages). Default is false.
input.sources	{object}	An array of objects, one for each input file.
input.sources[n].pages	{string}	The page numbers and/or page ranges, separated by commas, of the source document to convert.
processId	{string}	The id of the contentConverter resource which represents the file conversion operation.
state	{string}	The current state of the conversion process, which will be one of the following: "processing", "complete", or "error". If "processing", the conversion is still in progress. If "complete", the conversion has completed successfully. If "error", the conversion failed due to a problem.
percentComplete	{integer}	An integer from 0 to 100 that indicates what percentage of the conversion is complete.
errorCode	{string}	An error code string if a problem occurred during the conversion process.
output.results	{object}	An array of objects, one for each output file created. The 0-based index of each output file will be used in the URL <code>GET /v2/viewingSessions/{viewingSessionId}/contentConverters/{processId}/results/{index}/file</code> to retrieve the contents of each file.
output.results[n].sources	{object}	An array of objects, one for each source file which contributed to this output file.
output.results[n].sources[n].pages	{string}	The page or pages used from the source file. This will be a string value using one-based indexing. For example, if the output file represents page 2 of the source document, pages would have a value of "2". If the output file represents all 20 pages of a source document, pages would have a value of "1-20".
output.results[n].pageCount	{integer}	The total number of pages in the output file.

Example

```
GET pas_base_url/v2/viewingSessions/ZGZhc2RmYXNkZmFzZGZkcw/contentConverters/mxivIVSw7UhtLlyWwt3QEA
```

Successful Response

When the converter process completes with no conversion failures:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    },
    "sources": [
      {
        "pages": ""
      }
    ]
  },
  "processId": "mxivIVSw7UhtLlyWwt3QEA",
  "state": "complete",
  "percentComplete": 100,
  "output": {
    "results": [
      {
        "fileId": "eOsJIqI8aHkxVW0yJug",
        "sources": [
          {
            "pages": "1-4"
          }
        ]
      }
    ]
  }
}
```

```

        "pageCount": 4
      }
    ]
  }
}

```

When the converter process completes with a failure:

```

HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    },
    "sources": [
      {
        "pages": ""
      }
    ]
  },
  "processId": "mxivIVSw7UhtLlyWwt3QEA",
  "state": "error",
  "percentComplete": 100,
  "errorCode": "CouldNotConvert",
  "output": {
    "results": [
      {
        "errorCode": "CouldNotConvertPage",
        "sources": [
          {
            "pages": "3"
          }
        ]
      }
    ]
  }
}

```

Error Responses

When a response with a status code of 580 is received check the response body for details:

```

HTTP/1.1 580 InternalError
Content-Type: application/json

{
  "errorCode": "InternalError"
}

```

When a converter process does not exist for the given `processId`:

```

HTTP/1.1 404 Not Found

```

GET /v2/viewingSessions/{viewingSessionId}/contentConverters/{processId}/results/{index}/file?ContentDispositionFilename={fileName}

Routes key: `GetContentConverterOutputFileV2`

Gets the contents of a file produced by the converter process. This URL has the following parameters:

Parameter	Description
<code>{viewingSessionId}</code>	The ID provided in the response from <code>POST /ViewingSession</code> .
<code>{processId}</code>	The ID provided in the response from <code>POST /v2/viewingSessions/{viewingSessionId}/contentConverters</code> .
<code>{index}</code>	The 0-based index of the result file, originating from its position in the <code>response.output.results</code> array in the response from <code>GET /v2/viewingSessions/{viewingSessionId}/contentConverters/{processId}</code> .
<code>{fileName}</code>	The filename as a URL-encoded string, without extension, to be used in the <code>Content-Disposition</code> response header (the appropriate file extension such as <code>pdf</code> will be appended automatically).

Response Headers

Name	Description
<code>Content-Disposition</code>	Specifies 'attachment' disposition, RFC-2183 compatible <code>filename</code> parameter and an RFC-8187 compatible <code>filename*</code> parameter, allowing the use of non-ASCII filenames.
<code>Content-Type</code>	The most-specific MIME type for the returned document.

Response Body

The raw bytes of the result document.

Example

```
GET pas_base_url/v2/viewingSessions/ZGZhc2RmYXNkZmFzZGZkcw/contentConverters/mxivIVSw7UhtLlyWwt3QEA/results/0/file?
ContentDispositionFilename=MyFile
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/pdf
Content-Disposition: attachment; filename="Greek____.pdf"; filename*=UTF-8'Greek%CE%91%CE%92%CE%93%CE%94.pdf

<<PDF data>>
```

Error Responses

When an output file does not exist for the given `index`:

```
HTTP/1.1 404 Not Found
```

Content Converters (Deprecated)

Introduction

NOTE: The following URLs have been **deprecated** and will be removed from the public API in a future release. They are not available at all in PrizmDoc Cloud. **Please use the newer Content Converters API instead.**

Deprecated URLs

URL	Description
(DEPRECATED) POST /contentConverters	Used to start a conversion process for a particular document based on the viewing session. Use POST /v2/viewingSessions/{viewingSessionId}/contentConverters instead.
(DEPRECATED) GET /contentConverters/{processId}	Used to get the status of the specific conversion task. Use GET /v2/viewingSessions/{viewingSessionId}/contentConverters/{processId} instead.
(DEPRECATED) GET /WorkFile/{fileId}	Used to get the output PDF. Use GET /v2/viewingSessions/{viewingSessionId}/contentConverters/{processId}/results/{index}/file instead.

POST /contentConverters (Deprecated)

Deprecated. Use [POST /v2/viewingSessions/{viewingSessionId}/contentConverters](#) instead.

Converts a viewing session to a PDF.

Note that this URL is only available with PrizmDoc Viewer Self-Hosted.

Routes key: `CreateContentConverter`

Starts a conversion process for a particular document based on the viewing session.

```
POST http://localhost:3000/contentConverters
Content-Type: application/json

{ "viewingSessionId": "1234" }
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    },
    "sources": [
      {
        "fileId": "9BgnvnYFK96E0YOsK-A9xA",
        "pages": ""
      }
    ]
  },
  "expirationDateTime": "2015-11-04T19:20:09.280Z",
  "processId": "mxivIVSw7UhtLlyWwt3QEA",
  "state": "processing",
  "percentComplete": 0,
  "affinityToken": "wxyz"
}
```

Error Responses

When an unknown error occurs while gathering data:

```
HTTP/1.1 580 Server Error
Content-Type: application/json

{
  "errorCode": "InternalError"
}
```

GET /contentConverters/{processId} (Deprecated)

Routes key: `PollContentConverter`

Deprecated. Use [GET /v2/viewingSessions/{viewingSessionId}/contentConverters/{processId}](#) instead.

Gets the status of the process which is converting the viewing session to a PDF.

```
GET http://localhost:3000/contentConverters/9BgnvnYFK96E0YOsK-A9xA
Accusoft-Affinity-Token: wxyz
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "input": {
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    },
    "sources": [
      {
        "fileId": "9BgnvnYFK96E0YOsK-A9xA",
        "pages": ""
      }
    ]
  },
  "expirationDateTime": "2015-11-04T19:20:09.280Z",
  "processId": "mxivIVSw7UhtLlyWwt3QEA",
  "state": "complete",
  "percentComplete": 100,
  "output": {
    "results": [
      {
        "fileId": "eOsJIqI8aHkxVV0yJug",
        "sources": [
          {
            "fileId": "9BgnvnYFK96E0YOsK-A9xA",
            "pages": "1-4"
          }
        ],
        "pageCount": 4
      }
    ]
  }
}
```

Error Responses

When an unknown error occurs while gathering data:

```
HTTP/1.1 500 Server Error
Content-Type: application/json
```

```
{
  "errorCode": "InternalServerError"
}
```

GET /WorkFile/{fileId}?ContentDispositionFilename={file name}&affinityToken={affinityToken} (Deprecated)

Deprecated. Use [GET /v2/viewingSessions/{viewingSessionId}/contentConverters/{processId}/results/{index}/file](#) instead.

Used to get the output PDF.

Note that this URL is only available with PrizmDoc Viewer Self-Hosted.

Routes key: `GetWorkFile`


```
GET http://localhost:3000/WorkFile/eOsJIqI8aHkxVV0yJug?
ContentDispositionFilename=MyFile&affinityToken=wxyz
```

Successful Response

```
HTTP/1.1 200 OK
Content-Disposition: attachment; filename={documentDisplayName}.{ext}
Content-Type: {content type of the specific document}
```

Form Definitions

Introduction

The *form definitions* REST API allows an application to manage form definitions used by our [e-signature viewer](#).

Available URLs

URL	Description
GET /FormDefinitions	Gets the list of forms available on the server.
GET /FormDefinitions/{formDefinitionId}	Gets a specific form definition from the server.
POST /FormDefinitions	Creates a new form definition using the uploaded data.
POST /FormDefinitions/{formDefinitionId}	Updates an existing form definition with the new uploaded data. This will overwrite all of the existing data with the new uploaded data.
DELETE /FormDefinitions/{formDefinitionId}	Deletes a form definition from the server.

GET /FormDefinitions

Routes key: `GetFormDefinitions`

Gets the list of forms available on the server.

```
GET pas_base_url/FormDefinitions
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

[ {
  "name": "Form 1",
  "formRoles": {
```

```
    "formRole1": {
      "formRoleId": "formRole1",
      "fieldColor": "#439fe0",
      "displayName": "one",
      "sortIndex": 1
    },
    "formRole2": {
      "formRoleId": "formRole2",
      "fieldColor": "#58bb63",
      "displayName": "two",
      "sortIndex": 2
    }
  },
  "formDefinitionId": "03f3e9c4a976419da576276acc427700"
}, {
  "name": "Form 2",
  "formRoles": {},
  "formDefinitionId": "04a6032f3eaa4a8a9eb1b5fcelcb99e9"
}]
```

Error Responses

When an unknown error occurs while gathering data:

```
HTTP/1.1 500 Server Error
Content-Type: application/json

{
  "errorCode": "InternalError"
}
```

GET /FormDefinitions/{formDefinitionId}

Routes key: `GetFormDefinition`

Gets a specific form definition from the server.

```
GET pas_base_url/FormDefinitions/03f3e9c4a976419da576276acc427700
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "templateDocumentId": "Form 1.pdf",
  "globalSettings": { ... global settings ... },
  "formRoles": { ... form roles ... },
  "groups": {},
  "formName": "Form 1",
  "formData": [ ... form data ... ]
}
```

Error Responses

When the form definition does not exist:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  "errorCode": "NotFound"
}
```

When an unknown error occurs while gathering data:

```
HTTP/1.1 500 Server Error
Content-Type: application/json

{
  "errorCode": "InternalServerError"
}
```

POST /FormDefinitions

Routes key: `CreateFormDefinition`

Creates a new form definition using the uploaded data.

```
POST pas_base_url/FormDefinitions
Content-Type: application/json

{
  "templateDocumentId": "Form 3.pdf",
  "globalSettings": { ... global settings ... },
  "formRoles": { ... form roles ... },
  "groups": {},
  "formName": "Form 3",
  "formData": [ ... form data ... ]
}
```

Successful Response

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "formDefinitionId": "5418c96283bc469783bd30e7c8fdc059"
}
```

Error Responses

When an unknown error occurs while gathering data:

```
HTTP/1.1 500 Server Error
Content-Type: application/json

{
  "errorCode": "InternalServerError"
}
```

POST /FormDefinitions/{formDefinitionId}

Routes key: UpdateFormDefinition

Updates an existing form definition with the new uploaded data. This will overwrite all of the existing data with the new uploaded data.

```
POST pas\_base\_url/FormDefinitions/03f3e9c4a976419da576276acc427700
Content-Type: application/json

{
  "templateDocumentId": "Form 1.pdf",
  "globalSettings": { ... global settings ... },
  "formRoles": { ... form roles ... },
  "groups": {},
  "formName": "Form 1 - updated",
  "formData": [ ... form data ... ]
}
```

Successful Response

```
HTTP/1.1 200 OK
```

Error Responses

When the form definition does not exist:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  "errorCode": "NotFound"
}
```

When an unknown error occurs while gathering data:

```
HTTP/1.1 500 Server Error
Content-Type: application/json
```

```
{
  "errorCode": "InternalError"
}
```

DELETE /FormDefinitions/{formDefinitionId}

Routes key: `DeleteFormDefinition`

Deletes a form definition from the server.

```
DELETE pas_base_url/FormDefinitions/03f3e9c4a976419da576276acc427700
```

Alternatively, the `POST` method is supported for this request in combination with an `X-HTTP-Method-Override` header, as such:

```
POST pas_base_url/FormDefinitions/03f3e9c4a976419da576276acc427700
X-HTTP-Method-Override: DELETE
```

Successful Response

```
HTTP/1.1 204 No Content
```

Error Responses

When the form definition does not exist:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  "errorCode": "NotFound"
}
```

When an unknown error occurs while gathering data:

```
HTTP/1.1 580 Server Error
Content-Type: application/json

{
  "errorCode": "InternalError"
}
```

Form Extractors

Introduction

The *form extractors* REST API is used by our [e-signature viewer](#) to automatically detect form field elements in a document being viewed.

A *form extractor* resource represents an asynchronous form extraction process. Each *form extractor* that is created is assigned a unique `processId`.

Available URLs

URL	Description
GET /ViewingSession/u{viewingSessionId}/FormInfo	Returns what kind of form field data, if any, is available in a viewing session's source document.
POST /v2/viewingSessions/{viewingSessionId}/formExtractors	Creates a new <i>form extractor</i> from the source document of a viewing session, starting the process of extracting form field data.
GET /v2/viewingSessions/{viewingSessionId}/formExtractors/{processId}	Gets the status and final output of a <i>form extractor</i> created for a specified viewing session.

Output Schemas

- `"acroform"` Output
- `"rasterForm"` Output

GET /ViewingSession/u{viewingSessionId}/FormInfo

Returns what kind of form field data, if any, is available in a viewing session's source document.

Request

URL Parameters

Parameter	Description
<code>{viewingSessionId}</code>	The <code>viewingSessionId</code> which identifies the viewing session. Note this particular URL requires a letter 'u' to be provided before the <code>viewingSessionId</code>.

Successful Response

Response Body

JSON with information about what kind of form data, if any, is available in the source document of the viewing session.

- `formType[]` (Array of strings) Array of values indicating what types of form data, if any, are available for extraction from this viewing session's source document. Values will be one of the following:
 - `"acroform"` - The source document is a PDF which contains AcroForm data. The data can be

extracted by using an `input.formType` of `"acroform"` in a subsequent POST to create a *form extractor* process.

- `"xfa"` - The source document is a PDF which contains XFA form data. We do not yet support extraction of XFA data.
- `"rasterForm"` - The source document is a raster file which may or may not contain detectable form fields. You can attempt to extract form data by using an `input.formType` of `"rasterForm"` in a subsequent POST to create a *form extractor* process.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404		No viewing session with the provided <code>{viewingSessionId}</code> could be found.
480	<code>"DocumentNotProvidedYet"</code>	A source document has not been provided to the viewing session.
480	<code>"FeatureNotLicensed"</code>	You are not licensed to use the form extraction feature.
501	<code>"NotImplemented"</code>	Form extraction is not yet implemented for a viewing session which uses a cached viewing package.
580	<code>"InternalError"</code>	The server encountered an internal error when handling the request.

Example

Request

```
GET pas_base_url/ViewingSession/uXYZ.../FormInfo
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "formType": ["acroform"]
}
```

POST `/v2/viewingSessions/{viewingSessionId}/formExtractors`

Creates a new *form extractor* from the source document of a viewing session, starting the process of extracting form field data.

Request

Request Headers

Name	Description
Content-Type	Must be <code>application/json</code>

Request Body

- `input`
 - `password` (String) Password to open the source document, if required.
 - `formType` (String) **Required.** Type of form field data to extract. Must be one of the following:
 - `"acroform"` - Extract AcroForm field data from a PDF and return results in our `"acroform"` JSON format.
 - `"rasterForm"` - Detect visible form fields in a raster document and return results in our `"rasterForm"` JSON format.
- `minSecondsAvailable` (Integer) The minimum number of seconds this process will remain available to GET its status. The actual lifetime may be longer.

Successful Response

Response Body

JSON with metadata about the created *form extractor* process. You can check on the status of the form extraction process with additional [GET requests](#).

- `input` (Object) Input we accepted to create the *form extractor* process.
- `processId` (String) Unique id for the newly-created *form extractor* process.
- `state` (String) State of extracting form field data:
 - `"processing"` - The server is extracting form field data.
 - `"complete"` - All form field data has been extracted.
 - `"error"` - There was a problem extracting form field data.
- `percentComplete` (Integer) Percentage of form extraction which has completed (from 0 to 100).
- `expirationDateTime` (String) Currently planned date and time when the *form extractor* resource will expire and no longer be available. This time may be extended if we have need to keep using the data. Format is [RFC 3339 Internet Date/Time profile of ISO 8601], e.g. `"2016-11-05T08:15:30.494Z"`.
- `errorCode` (String) Descriptive error code. Present when `state` is `"error"`.
- `errorDetails` (Object) Additional error details, if any. May be present when `errorCode` is present.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
480	<code>"MissingInput"</code>	A required input value was not provided. See <code>errorDetails</code> in the response body.
480	<code>"InvalidInput"</code>	An invalid input value was used. See <code>errorDetails</code> in the response body.
480	<code>"DocumentNotProvidedYet"</code>	A source document has not been provided to the viewing session.
480	<code>"FeatureNotLicensed"</code>	You are not licensed to use the form extraction feature.
480	<code>"LicenseCouldNotBeVerified"</code>	The server's license could not be verified. If you are evaluating the product without a license, the product is running in evaluation mode and this particular part of the product is unavailable without a license. If you have a license, make sure you configured your license

Status Code	JSON <code>errorCode</code>	Description
		correctly, that your license has not expired, and that you have not exceeded any license limits (such as, for a Cloud License, the total number of logical CPU cores in use).
501	"NotImplemented"	Form extraction is not yet implemented for a viewing session which uses a cached viewing package.
580	"InternalError"	The server encountered an internal error when handling the request.

Example

Request

```
POST pas_base_url/v2/viewingSessions/uXYZ.../formExtractors
Content-Type: application/json

{
  "input": {
    "formType": "acroform"
  }
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "formType": "acroform"
  },
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "state": "processing",
  "percentComplete": 0,
  "expirationDateTime": "2016-12-17T20:38:39.796Z"
}
```

GET

`/v2/viewingSessions/{viewingSessionId}/formExtractors/{processId}`

Gets the status and final output of a *form extractor* created for a specified viewing session.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.
{processId}	The <code>processId</code> which identifies the <i>form extractor</i> process.

Successful Response

Response Body

JSON with metadata about the *form extractor* process and the final `output`, if available. You can check on the status of the form extraction process with additional GET requests.

- `input` (Object) Input we accepted to create the form extraction process.
- `processId` (String) Unique id for this *form extractor* process.
- `state` (String) State of extracting form field data:
 - "processing" - The server is extracting form field data.
 - "complete" - All form field data has been extracted.
 - "error" - There was a problem extracting form field data.
- `percentComplete` (Integer) Percentage of form extraction which has completed (from 0 to 100).
- `expirationDateTime` (String) Currently planned date and time when the *form extractor* resource will expire and no longer be available. This time may be extended if we have need to keep using the data. Format is [RFC 3339 Internet Date/Time profile of ISO 8601], e.g. "2016-11-05T08:15:30.494Z".
- `errorCode` (String) Descriptive error code. Present when `state` is "error".
- `errorDetails` (Object) Additional error details, if any. May be present when `errorCode` is present.
- `output` (Object) Present when `state` is "complete":
 - `acroform` (Object) Present when `input.formType` is "acroform". See "`acroform`" [Output](#) below for details.
 - `rasterForm` (Object) Present when `input.formType` is "rasterForm". See "`rasterForm`" [Output](#) below for details.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No <i>form extractor</i> could be found for the given {viewingSessionId} and {processId}.
501	"NotImplemented"	Form extraction is not yet implemented for a viewing session which uses a cached viewing package.
580	"InternalError"	The server encountered an internal error when handling the request.

Examples

Request

```
GET pas_base_url/v2/viewingSessions/uXYZ.../formExtractors/x62gH3TYdqlKj94pLqzmtS
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "formType": "acroform"
  },
  "output": {
    "acroform": {
      "pages": [
        {
          "page": 1,
          "height": 792,
          "width": 612,
          "fields": [
            {
              "fieldType": "Text",
              "name": "email",
              "required": true,
              "tabOrder": 0,
              "appearance": {
                "textColor": "0 g",
                "font": "Helvetica"
              },
              "boundingBox": {
                "lowerLeftX": 89,
                "lowerLeftY": 646,
                "upperRightX": 239,
                "upperRightY": 668
              },
              "options": {
                "multiline": false,
                "maxLen": -1
              },
              "format": {
                "formatCategory": "None"
              }
            },
            {
              "fieldType": "Text",
              "name": "fullName",
              "required": false,
              "tabOrder": 1,
              "appearance": {
                "textColor": "0 g",
                "font": "Helvetica"
              },
              "boundingBox": {
                "lowerLeftX": 89,
                "lowerLeftY": 676,
                "upperRightX": 239,
                "upperRightY": 698
              },
              "options": {
                "multiline": false,
                "maxLen": -1
              },
              "format": {
                "formatCategory": "None"
              }
            }
          ]
        }
      ]
    }
  }
}
```

```

    ]
  }
]
},
"expirationDateTime": "2016-10-11T03:30:33.166Z",
"percentComplete": 100,
"processId": "x62gH3TYdqlKj94pLqzmtS",
"state": "complete"
}

```

"acroform" Output

The `output.acroform` object will conform to the following. All properties are always present unless otherwise noted:

- `pages[]` (Array of Objects) Pages in the document which contains acroform fields. Array will be empty if document does not contain any acroform fields. Each item will contain:
 - `page` (Integer) One-indexed page number.
 - `height` (Number) Page height in points.
 - `width` (Number) Page width in points.
 - `fields[]` (Array of Objects) Acroform fields in the current page. Items may contain:
 - `fieldType` (String) Field type. Will be one of the following:
 - "Text" - Text field
 - "Button" - Push button, check box, or radio button:
 - push button when `options.pushButton` is true
 - check box when `options.pushButton` and `options.radio` are both false
 - radio button when `options.radio` is true
 - "Signature" - Signature field
 - `name` (String) Unique field or radio button group name.
 - `required` (Boolean) Indicates whether or not this field is required for the form to be considered complete.
 - `tabOrder` (Integer) Tab order of the field within the document.
 - `boundingBox` (Object) Position and size of this field. Object will contain:
 - `lowerLeftX` (Number) Distance in points from the left edge of the page to the left side of this field.
 - `lowerLeftY` (Number) Distance in points from the bottom edge of the page to the bottom edge of this field.
 - `upperRightX` (Number) Distance in points from the left edge of the page to the right edge of this field.
 - `upperRightY` (Number) Distance in points from the bottom edge of the page to the top edge of this field.
 - `appearance` (Object) Field appearance details:
 - `textColor` (String) Text fill color. Not always present.
 - `font` (String) Font name to use for this field. Not always present.
 - `format` (Object) Field formatting details:
 - `formatCategory` (String) Will be one of the following:
 - "None" - Indicates there are no additional `formatOptions` for this field.
 - "Date" - For text fields, requires the field value to be a date.
 - `formatOptions` Additional options for the given `formatCategory`, if any:
 - When `formatCategory` is "Date": (String) **Date format string** to use when formatting the date value for display.

- `options` (Object) Additional field options, present for some field types:
 - When `fieldType` is `"Text"`:
 - `multiline` (Boolean) Indicates whether or not this is a multi-line text field.
 - `maxLen` (Integer) Indicates the maximum number of characters this form field accepts, or `-1` if there is no limit.
 - When `fieldType` is `"Button"`:
 - `pushButton` (Boolean) `true` if this field is a push button, `false` otherwise.
 - `radio` (Boolean) `true` if this field is a radio button, `false` otherwise.
 - When both `pushButton` and `radio` are `false`, this field is a check box.
 - When `fieldType` is `"Button"` and `options.radio` is `true`:
 - `buttonOnValue` (String) Indicates the form value to use when this radio button is selected.
 - `buttonOffValue` (String) Indicates the form value to use when this radio button is not selected. Value will always be `"Off"`.
 - `buttonValue` (String) Indicates whether or not this radio button should be initially selected. When the value matches `buttonOnValue`, then this radio button should be initially selected. Otherwise (when the value is `"Off"`), this radio button should not be initially selected.

Fill Color Strings

A string of one or more numbers followed by an operator indicating what the numbers represent:

- Grayscale value (when string ends in `"g"`): A single number between 0 and 1 followed by `"g"` represents the amount of white which forms a grayscale color value. For example:
 - `"0 g"` - black
 - `"0.5 g"` - 50% gray
 - `"1 g"` - white
- RGB value (when string ends in `"rg"`): Three numbers between 0 and 1 followed by `"rg"` represent the the amount of red, green, and blue light which are additively mixed to form the final color. For example:
 - `"1 0 0 rg"` - red
 - `"1 1 0 rg"` - yellow
 - `"0.5 0.25 0.75 rg"` - 50% red, 25% blue, 75% green
- CMYK (when string ends in `"k"`): Four numbers between 0 and 1 followed by `"k"` represent the amount of cyan, magenta, yellow, and black which should be subtractively mixed to form the final color. For example:
 - `"0 0 0 1 k"` - black
 - `"1 1 1 0 k"` - black
 - `"1 1 1 1 k"` - black
 - `"1 0 0 0 k"` - cyan
 - `"0.25 0.88 0.2 0.16 k"` - 25% cyan, 88% magenta, 20% yellow, 16% black

Date Format Strings

Date format strings use the following special substitution patterns:

- `yy` - 2-digit year (e.g. `16` for the year 2016)
- `yyyy` - 4-digit year (e.g. `2016`)
- `m` - Month number with no zero padding (e.g. `7` for July)
- `mm` - Month number zero-padded to always be two characters long (e.g. `07` for July)
- `mmm` - Abbreviated month name (e.g. `Jan`)
- `mmmm` - Full month name (e.g. `January`)
- `d` - Day of the month with no zero padding (e.g. `4` for the fourth day of the month)
- `dd` - Day of the month zero-padded to always be two characters (e.g. `04` for the fourth day of the month)

- `ddd` - Abbreviated day of the week (e.g. `Sun`)
- `dddd` - Full name for the day of the week (e.g. `Sunday`)
- `h` - Hour number in 12-hour time with no zero padding (e.g. `2` for 2 o'clock)
- `hh` - Hour number in 12-hour time zero-padded to always be two characters (e.g. `02` for 2 o'clock)
- `H` - Hour number in 24-hour time with no zero padding (e.g. `13` for the 1:00 pm hour)
- `HH` - Hour number in 24-hour time zero-padded to always be two characters (e.g. `02` for the 2:00 am hour)
- `M` - Minute without zero padding
- `MM` - Minute, zero-padded to always be two digits
- `s` - Second without zero-padding
- `ss` - Second, zero-padded to always be two digits
- `z` - Offset from UTC (e.g. `-0400`)
- `j` - Abbreviated Japanese era and year (e.g. `H28` for the year 2016).
- `jj` - Full Japanese era and year (e.g. `平成28` for the year 2016).
- `jjj` - Japanese era year without specifying the era (e.g. `28` for the year 2016).

All other characters are considered literal punctuation for the format string. The special characters used above may be used literally by escaping them with a backslash.

"rasterForm" Output

The `output.rasterForm` object will conform to the following. All properties are always present unless otherwise noted:

- `pages[]` (Array of Objects) Information about each page in the raster document. Each item will contain:
 - `page` (Integer) One-indexed page number.
 - `height` (Number) Page height in pixels.
 - `width` (Number) Page width in pixels.
 - `fields[]` (Array of Objects) Fields detected in the current page. Array will be empty if no fields were detected. Items will contain:
 - `name` (String) Unique name we have automatically assigned to this field in the document (e.g. `"field5"`).
 - `fieldType` (String) Field type. Will be one of the following:
 - `"Text"` - Text field
 - `"CheckBox"` - Check box
 - `confidence` (Number) Our confidence in the correct detection of this field using a scale of 0 (no confidence) to 100 (complete confidence).
 - `boundingBox` (Object) Position and size of this field. Object will contain:
 - `x` (Number) Distance in pixels from the left edge of the page to the left side of this field.
 - `y` (Number) Distance in pixels from the top edge of the page to the top edge of this field.
 - `width` (Number) Distance in pixels from the left edge of this field (`x`) to the right edge of this field.
 - `height` (Number) Distance in pixels from the top edge of this field (`y`) to the bottom edge of this field.
 - `tables[]` (Array of Objects) Tables detected in the current page. Array will be empty if no tables were detected. Items will contain:
 - `numOfColumns` (Integer) Number of columns in the detected table.
 - `numOfRows` (Integer) Number of rows in the detected table.
 - `fields[]` (Array of Objects) Fields detected in the current table. Items will contain:
 - `name` (String) Unique name we have automatically assigned to this field in the document (e.g. `"field5"`).
 - `fieldType` (String) Field type. Will be one of the following:
 - `"Text"` - Text field

- "CheckBox" - Check box
- `confidence` (Number) Our confidence in the correct detection of this field using a scale of 0 (no confidence) to 100 (complete confidence).
- `boundingBox` (Object) Position and size of this field. Object will contain:
 - `x` (Number) Distance in pixels from the left edge of the page to the left side of this field.
 - `y` (Number) Distance in pixels from the top edge of the page to the top edge of this field.
 - `width` (Number) Distance in pixels from the left edge of this field (`x`) to the right edge of this field.
 - `height` (Number) Distance in pixels from the top edge of this field (`y`) to the bottom edge of this field.

Image Stamps

Introduction

The *image stamps* REST API is used by our viewer to load image stamps. It is unusual for your application to need to use this REST API.

By default, a PAS installation contains two image stamps: a green check and a red X. To customize your image stamps, see [Add Custom Image Stamps](#).

Available URLs

URL	Description
GET /ImageStampList	Gets list of available image stamps.
GET /ImageStamp/{imageStampId}/q?format={formatStr}	Gets image data for an image stamp.

GET /ImageStampList

Routes key: `GetImageStamps`

```
GET pas_base_url/ImageStampList
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  imageStamps : [
    { id: "ZmlsZTEuZ2lm", displayName: "file1.gif" },
    { id: "ZmlsZTIucG5n", displayName: "file2.png" }
  ]
}
```

GET /ImageStamp/{imageStampId}/q?format={formatStr}

Routes key: `GetImageStamp`

Gets image data for an image stamp. The query string parameter `format` defined the format of the response, and supports the following values:

- `Base64`: Returns a JSON object containing the image as a base64 encoded string, as well as a hash of the original image.
- `Image`: Returns the raw image file itself.

The raw image file itself is returned if no format or an unsupported format is specified.

Getting Base64 data

```
GET pas_base_url/ImageStamp/ZmlsZTIucG5n/q?format=Base64
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "dataHash": "1ca8d2e80b6f8f2774f3bc0e6422bc653b0e4d80",
  "dataUrl": "data:image/png;base64,..."
}
```

Getting the image data

```
GET pas_base_url/ImageStamp/ZmlsZTIucG5n/q?format=Image
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: {content type of the image}

<<binary image data>>
```

Error Responses

When an invalid `imageStampId` is requested:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
  "errorCode": "BadRequest"
}
```



```
}
```

When a valid `imageStampId` is requested but it does not exist:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  "errorCode": "NotFound"
}
```

Legacy Create Session

Introduction

DEPRECATED: The legacy *create session* REST API is deprecated in PAS and is available for backwards compatibility only. Please use the newer [viewing sessions](#) REST API instead.

Available URLs

URL	Description
GET /CreateSession?document={document name}	Creates a viewing session based on a specific document in storage.
GET /CreateSession?form={formDefinitionId}	Creates a viewing session for the document referenced by a form definition.

GET /CreateSession?document={document name}

Note that this URL is only available with PrizmDoc Viewer Self-Hosted.

Routes key: `LegacyCreateSession`

Creates a viewing session based on a specific document in storage.

```
GET pas_base_url/CreateSession?document=Sample.doc
```

To make the equivalent call using the new `ViewingSession` API:

```
POST pas_base_url/ViewingSession
Content-Type: application/json

{
  "source": {
    "type": "document",
    "fileName": "Sample.doc"
  }
}
```

```
}
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "viewingSessionId": "XYZ..."
}
```

Error Responses

When the document does not exist:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  "errorCode": "DocumentNotFound"
}
```

When an unknown error occurs while gathering data:

```
HTTP/1.1 500 Server Error
Content-Type: application/json

{
  "errorCode": "InternalError"
}
```

GET /CreateSession?form={formDefinitionId}

Note that this URL is only available with PrizmDoc Viewer Self-Hosted.

Routes key: `LegacyCreateSession`

Creates a viewing session for the document referenced by a form definition.

```
GET pas_base_url/CreateSession?form=03f3e9c4a976419da576276acc427700
```

There is no equivalent to this call in the new `ViewingSession` API. Instead, the user should refer to the `templateDocumentId` field in the form definition JSON object, and use it as a document to create a viewing session.

Error Responses

When the form definition does not exist:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  "errorCode": "FormDefinitionNotFound"
}
```

When the document referenced by the form definition does not exist:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  "errorCode": "DocumentNotFound"
}
```

When an unknown error occurs while gathering data:

```
HTTP/1.1 500 Server Error
Content-Type: application/json

{
  "errorCode": "InternalServerError"
}
```

Markup Burners

Introduction

The PAS *markup burners* REST API is used by our viewer to allow the user to download the document they are viewing as a PDF with all annotations applied.

This REST API is designed for our viewer. If your application needs to perform its own markup burning, **we recommend you use [PrizmDoc Server's markup burners REST API](#) instead.** It allows your application to produce PDFs with annotations "burned in" without the need for a viewing session.

Available URLs

URL	Description
POST /ViewingSession/u{viewingSessionId}/MarkupBurner	Starts a new MarkupBurner using the source document of a viewing session and provided markup data as input.
GET /ViewingSession/u{viewingSessionId}/MarkupBurner/{processId}	Gets the status of a MarkupBurner for a viewing session.
GET /ViewingSession/u{viewingSessionId}/MarkupBurner/{processId}/Document	Gets the output result of a MarkupBurner process for a viewing session.

POST /ViewingSession/u{viewingSessionId}/MarkupBurner

Routes key: `CreateMarkupBurner`

Starts a new MarkupBurner using the source document of a viewing session and a provided markup data as input. When the asynchronous process is ultimately finished, the output will be a new document which includes the provided markup as part of the document itself (the original source document of the viewing session is left unaltered).

This is a specialized URL which allows you to do markup burning against the source file of an existing viewing session without needing to use the work file API.

This request merely begins the markup burning process. Once started, you poll the status of the process using the [GET /ViewingSession/u{viewingSessionId}/MarkupBurner/{processId}](#) URL below to know when the process has completed.

Request

Request Headers

Name	Description
Content-Type	Specifies the type of content being provided to the markup burner process. It must be <code>application/xml</code> or <code>application/json</code> depending on the markup format used in the request body.

URL Parameters

Parameter	Description
{viewingSessionId}	The id provided in the response from POST /PCCIS/V1/ViewingSession .

Query String Parameters

Parameter	Description
{redactionMode}	How redactions should be applied. May be one of the following: <ul style="list-style-type: none"> "normal" - Actually redact the document, removing document content covered by redactions, drawing opaque redaction rectangles, and drawing any associated redaction reason text in the center of the rectangles. "draft" - Do NOT actually redact the document. Instead, indicate which parts of the document would be redacted by drawing partially transparent redaction rectangles over the parts of the document that would be redacted. In order to avoid interfering with the original document content, redaction reason text will not be drawn in the center of the transparent redaction rectangles. Default is "normal".
{redactionDraftOpacity}	Controls the opacity of redactions when <code>redactionMode</code> is set to "draft". Must be a value within 0 and 1 where 0 is fully transparent and 1 is fully opaque. Default is 0.2.
{RemoveFormFields}	Optional parameter indicating which interactive form fields to remove from the source document upon markup burner process. Currently only "acroform" value is supported. Please see example below.

Request Body

The JSON or XML markup to burn into the source document.

Response Body

If successful, a JSON object which may contain:

- `processId` (String) The id of the process.
- `state` (String) The current state of the markup burning process running on the server. This will always be **"processing"** in the initial POST response.
- `percentComplete` (Integer) The percentage (0 - 100) complete of the process. This will always be **0** in the initial POST response.
- `input` (null) Legacy property which exists only for backwards compatibility. Value will always be `null`.
- `output` (null) Legacy property which exists only for backwards compatibility. Value will always be `null`.

Error Responses

Status Code	Description
404	Not Found, if {viewingSessionId} does not exist.
405	Method Not Allowed, if POST HTTP method is not used.
480	An invalid input value was used. See <code>errorDetails</code> in the response body.

Examples

Request to burn a rectangle annotation using JSON markup

```
POST pas_base_url/ViewingSession/uDLbVh9sTmXJAmD1GeXbs9Gn3WHxs8oib2xPsW2xEfjnIDdoJcudPtxciodSYFQq6zYGabQ_rJIecdbkImTTkSA/MarkupBurner
Content-Type: application/json
```

```
{
  "marks": [
    {
      "uid": "Z2diOV8yMDE3LTAzLTMxVDA3OjQ5OjExLjUyNVpFY2VnNjZy",
      "interactionMode": "Full",
      "pageNumber": 1,
      "type": "RectangleAnnotation",
      "creationDateTime": "2017-03-31T07:49:11.525Z",
      "modificationDateTime": "2017-03-31T07:49:11.526Z",
      "data": {},
      "conversation": {},
      "rectangle": {
        "x": 0,
        "y": 0,
        "width": 0,
        "height": 0
      },
      "pageData": {
        "width": 612,
        "height": 792
      },
      "borderColor": "#000000",
      "borderThickness": 4,
      "fillColor": "#FB0404",
      "opacity": 255
    }
  ]
}
```

Request to burn a rectangle annotation using JSON markup with removal of acroform fields

```
POST prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uDLbVh9sTmXJAmD1GeXbs9Gn3WHxs8oib2xPsW2xEfjnIDdoJcudPtxciodSYFQq6zYGabQ_rJIecdbkImTTkSA/MarkupBurner?
RemoveFormFields=acroform
Content-Type: application/json
```

```
{
  "marks": [
    {
      "uid": "Z2diOV8yMDE3LTAzLTMxVDA3OjQ5OjExLjUyNVpFY2VnNjZy",
      "interactionMode": "Full",
      "pageNumber": 1,
      "type": "RectangleAnnotation",
      "creationDateTime": "2017-03-31T07:49:11.525Z",
      "modificationDateTime": "2017-03-31T07:49:11.526Z",
      "data": {},
      "conversation": {},
      "rectangle": {
        "x": 0,
        "y": 0,
        "width": 0,
        "height": 0
      },
      "pageData": {
        "width": 612,
        "height": 792
      },
      "borderColor": "#000000",
```

```

    "borderThickness": 4,
    "fillColor": "#FB0404",
    "opacity": 255
  }
}
}

```

Response

```

HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": null,
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "state": "processing",
  "percentComplete": 0,
  "errorCode": null,
  "output": null
}

```

GET /ViewingSession/u{viewingSessionId}/MarkupBurner/{processId}

Routes key: PollMarkupBurner

Gets the status of a MarkupBurner for a viewing session.

Requests are typically sent to this URL repeatedly as long as the `state` is "processing".

When `state` is "complete", a new document with the provided markup burned into it will be available at:

```
GET /ViewingSession/u{viewingSessionId}/MarkupBurner/{processId}/Document
```

If an error occurred and the output could not be created, the `state` property will be "error" and the `errorCode` property will contain an error code string.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The id provided in the response from POST /ViewingSession.
{processId}	The id of the process.

Response Body

If successful, a JSON object which may contain:

- `processId` (String) The id of the process.
- `state` (String) The current state of the process. The following values are allowed:
 - "processing" - The markup burning is in progress.
 - "complete" - The markup burning is completed.
 - "error" - The markup burning returns an error.
- `percentComplete` (Integer) The percentage (0 - 100) complete of the process.
- `errorCode` (String) An error code string if a problem occurred during processing.
- `input` (null) Legacy property which exists only for backwards compatibility. Value will always be `null`.
- `output` (null) Legacy property which exists only for backwards compatibility. Value will always be `null`.

Error Responses

Status Code	Description
404	Not Found, if either the {viewingSessionId} or {processId} do not exist.
405	Method Not Allowed, if GET method is not used.

Examples

Request

```

GET
pas_base_url/PCCIS/V1/ViewingSession/uDLbVh9sTmXJAmd1GeXbS9Gn3WHxs8oib2xPsw2xEFjnIDdoJcudPtxciodySFQq6zYGabQ_rJTeedbkImTTkSA/MarkupBurner/5rGUUh3Qxhf6VXm8RkBPfA

```

Response when the state is "processing"

```

HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": null,
  "processId": "5rGUUh3Qxhf6VXm8RkBPfA",
  "state": "processing",
  "percentComplete": 0,
  "errorCode": null,
  "output": null
}

```

Response when the state is "complete"

```

HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": null,
  "processId": "5rGUUh3Qxhf6VXm8RkBPfA",
  "state": "complete",
  "percentComplete": 100,
}

```

```
"errorCode": null,
"output": null
}
```

Response when the state is "error" because the provided markup XML did not contain valid XML markup or the provided markup JSON was auto-detected as a non-text format

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": null,
  "processId": "53LckvO8-rsDIAw95WgoFA",
  "state": "error",
  "percentComplete": 0,
  "errorCode": "RedactionError",
  "output": null
}
```

Response when the state is "error" because the provided markup JSON cannot be parsed as JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": null,
  "processId": "53LckvO8-rsDIAw95WgoFA",
  "state": "error",
  "percentComplete": 0,
  "errorCode": "InvalidJson",
  "output": null
}
```

Response when the state is "error" because the provided markup JSON does not match the JSON Marks Schema

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": null,
  "processId": "53LckvO8-rsDIAw95WgoFA",
  "state": "error",
  "percentComplete": 0,
  "errorCode": "InvalidMarkup",
  "output": null
}
```

GET /ViewingSession/{viewingSessionId}/MarkupBurner/{processId}/Document?ContentDispositionFilename={ContentDispositionFilename}

Routes key: `GetBurnedDocument`

Downloads the resulting burned-in document. It will be served with the correct `Content-Type` of the document, and using the specified `ContentDispositionFilename` as the name, along with the correct file extension.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The id provided in the response from POST /ViewingSession.
{processId}	The id of the process which identifies the MarkupBurner task as a string.
{ContentDispositionFilename}	The filename as a URL-encoded string, without extension, to be used in the <code>Content-Disposition</code> response header (the file extension will be appended automatically). The default value is "document".

Response Headers

Name	Description
Content-Disposition	Specifies 'attachment' disposition, RFC-2183 compatible <code>filename</code> parameter and an RFC-8187 compatible <code>filename*</code> parameter, allowing the use of non-ASCII filenames.
Content-Type	Will be <code>application/pdf</code> .

Response Body

The raw bytes of the PDF document with markup burned into it.

Error Responses

Status Code	Description
404	Not Found, which may occur if any of the following are true: the MarkupBurner has not completed yet, no such MarkupBurner exists, or no such ViewingSession exists.
405	Method Not Allowed, if GET method is not used.

Examples

Request

```
GET
pas_base_url/ViewingSession/uDLbVh9sTmXJAmDlGeXbS9Gn3WHxs8oib2xPsW2xEfjnIDdoJcudPtxciodSYPQq6zYGabQ_rJIecdbkImTTkSA/MarkupBurner/ElkNzWtrUJp4rXI5YnLUgw/Document
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/pdf
Content-Disposition: attachment; filename="Greek____.pdf"; filename*=UTF-8''Greek%CE%91%CE%92%CE%93%CE%94.pdf
<<PDF data>>
```

Markup Layers

Introduction

The *markup layers* REST API is used by our viewer to save and load markup (annotations and redactions) in our newer JSON format.

Background

Our viewer has two different markup persistence modes it operates in:

- **Legacy XML markup mode** - In this mode, the viewer saves and loads markup data in our older, legacy XML format using the *markup XML* REST API. This is the default viewer behavior.
- **JSON markup mode** - In this mode, the viewer saves and loads markup data in our newer JSON format using this *markup layers* REST API. This mode applies when the viewer is configured with `annotationsMode` set to `'LayeredAnnotations'` (recommended).

We recommend you always instantiate the viewer with `annotationsMode` set to `'LayeredAnnotations'` so that it will operate in the new JSON markup mode. When you do this, the viewer will use this *markup layers* REST API to save and load markup data in our newer JSON format.

If your application needs to access or manipulate the annotation JSON created by your end users, you will need to use this REST API.

A set of markup data is accessed by `markupId`. But, because this REST API was designed for use by our viewer, all operations are based upon a *viewing session*. In order to get access to a set of markup data by `markupId`, you first must create a “dummy” viewing session, specifying the `markupId` you want to access via the `source.markupId` property of your `POST /ViewingSession` request.

However, because `source.markupId` is optional when creating a viewing session, you may not know the `markupId` to use. Fortunately, as long as you still know the information about the source document and how you created the original viewing session for your end user, there is still a way to access the markup data. When a viewing session is created without a `source.markupId` specified, we will automatically calculate a `markupId` value *based upon the other source object properties*. So, if you did not specify an explicit `source.markupId` when creating the original viewing session for your end user, you can still access the markup data that end user saved by creating a new “dummy” viewing session with exactly the same set of `source` properties. For more information about the `source` object options you can pass in on a `POST /ViewingSession` request, [see the `POST /ViewingSession` API reference](#).

Available URLs

URL	Description
GET /MarkupLayers/u{viewingSessionId}	Gets the list of all available markup layers for the particular document.
POST /MarkupLayers/u{viewingSessionId}	Creates a new markup layer using the provided data.

URL	Description
GET /MarkupLayers/u{viewingSessionId}/{layerRecordId}	Gets a particular layer record from the server.
PUT /MarkupLayers/u{viewingSessionId}/{layerRecordId}	Updates an existing markup layer record with new data. This will overwrite all old data with the new uploaded data.
DELETE /MarkupLayers/u{viewingSessionId}/{layerRecordId}	Deletes a markup layer record from the server.

GET /MarkupLayers/u{viewingSessionId}

Routes key: `GetMarkupLayers`

Gets the list of all available markup layers for the particular document.

```
GET pas_base_url/MarkupLayers/u1234
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "name": "layer name 1",
    "layerRecordId": "2fee93fadf2ed11df",
    "originalXmlName": ""
  },
  {
    "name": "layer name 2",
    "layerRecordId": "32f993b09fb0f2236",
    "originalXmlName": ""
  }
]
```

Error Responses

When the document cannot be identified based on the viewing session:

```
HTTP/1.1 502 Bad Gateway
Content-Type: application/json

{
  "errorCode": "BadGateway"
}
```

When the server's license could not be verified:

```
HTTP/1.1 480 LicenseCouldNotBeVerified
```



```
Content-Type: application/json

{
  "errorCode": "LicenseCouldNotBeVerified"
}
```

When an unknown error occurs while gathering data:

```
HTTP/1.1 500 Server Error
Content-Type: application/json

{
  "errorCode": "InternalServerError"
}
```

POST /MarkupLayers/u{viewingSessionId}

Routes key: `CreateMarkupLayer`

Creates a new markup layer using the provided data.

```
POST pas_base_url/MarkupLayers/u1234
Content-Type: application/json

{
  "name": "layer name 1",
  "comments": [],
  "data": {},
  "marks": [{ array of mark objects }]
}
```

Successful Response

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "layerRecordId": "2fee93fadf2ed11df"
}
```

Error Responses

When the document cannot be identified based on the viewing session:

```
HTTP/1.1 502 Bad Gateway
Content-Type: application/json

{
  "errorCode": "BadGateway"
}
```

When the server's license could not be verified:

```
HTTP/1.1 480 LicenseCouldNotBeVerified
Content-Type: application/json

{
  "errorCode": "LicenseCouldNotBeVerified"
}
```

When an unknown error occurs while gathering data:

```
HTTP/1.1 580 Server Error
Content-Type: application/json

{
  "errorCode": "InternalError"
}
```

GET /MarkupLayers/u{viewingSessionId}/{layerRecordId}

Routes key: `GetMarkupLayer`

Gets a particular layer record from the server.

```
GET pas_base_url/MarkupLayers/u1234/2fee93fadf2ed11df
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  ...markup layer data...
}
```

Error Responses

When the markup layer record does not exist:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  "errorCode": "NotFound"
}
```

When the document cannot be identified based on the viewing session:

```
HTTP/1.1 502 Bad Gateway
Content-Type: application/json

{
  "errorCode": "BadGateway"
}
```

When an unknown error occurs while gathering data:

```
HTTP/1.1 580 Server Error
Content-Type: application/json

{
  "errorCode": "InternalError"
}
```

PUT /MarkupLayers/u{viewingSessionId}/{layerRecordId}

Routes key: `UpdateMarkupLayer`

Updates an existing markup layer record with new data. This will overwrite all old data with the new uploaded data.

```
PUT pas_base_url/MarkupLayers/u1234/2fee93fadf2ed11df
Content-Type: application/json
```

Alternatively, the `POST` method is supported for this request in combination with an `X-HTTP-Method-Override` header, as such:

```
POST pas_base_url/MarkupLayers/u1234/2fee93fadf2ed11df
Content-Type: application/json
X-HTTP-Method-Override: PUT
```

Successful Response

```
HTTP/1.1 200 OK
```

Error Responses

When the markup layer record does not exist:

```
HTTP/1.1 404 Not Found
Content-Type: application/json
```

```
{
  "errorCode": "NotFound"
}
```

When the document cannot be identified based on the viewing session:

```
HTTP/1.1 502 Bad Gateway
Content-Type: application/json

{
  "errorCode": "BadGateway"
}
```

When an unknown error occurs while gathering data:

```
HTTP/1.1 500 Server Error
Content-Type: application/json

{
  "errorCode": "InternalServerError"
}
```

DELETE /MarkupLayers/u{viewingSessionId}/{layerRecordId}

Routes key: `DeleteMarkupLayer`

Deletes a markup layer record from the server.

```
DELETE pas_base_url/MarkupLayers/u1234/2fee93fadf2ed11df
```

Alternatively, the `POST` method is supported for this request in combination with an `X-HTTP-Method-Override` header, as such:

```
POST pas_base_url/MarkupLayers/u1234/2fee93fadf2ed11df
X-HTTP-Method-Override: DELETE
```

Successful Response

```
HTTP/1.1 204 No Content
```

Error Responses

When the markup layer record does not exist:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  "errorCode": "NotFound"
}
```

When the document cannot be identified based on the viewing session:

```
HTTP/1.1 502 Bad Gateway
Content-Type: application/json

{
  "errorCode": "BadGateway"
}
```

When an unknown error occurs while gathering data:

```
HTTP/1.1 580 Server Error
Content-Type: application/json

{
  "errorCode": "InternalError"
}
```

Markup XML

Introduction

The *markup XML* REST API is used by our viewer to save and load markup (annotations and redactions) in our legacy XML format.

Background

Our viewer has two different markup persistence modes it operates in:

- **Legacy XML markup mode** - In this mode, the viewer saves and loads markup data in our older, legacy XML format using this *markup XML* REST API. This is the default viewer behavior.
- **JSON markup mode** - In this mode, the viewer saves and loads markup data in our newer JSON format using the *markup layers* REST API. This mode applies when the viewer is configured with `annotationsMode` set to `'LayeredAnnotations'` (recommended).

We recommend you always instantiate the viewer with `annotationsMode` set to `'LayeredAnnotations'` so that it will operate in the new JSON markup mode. When you do this, the viewer will NOT use this REST API but will instead use this *markup layers* REST API to save and load markup data in our newer JSON format.

However, **if your application is already using the viewer in the legacy XML markup mode and you need to access or manipulate the markup XML created by your end users, you will need to use this REST API.**

A set of markup data is accessed by `markupId`. But, because this REST API was designed for use by our viewer, all

operations are based upon a *viewing session*. In order to get access to a set of markup data by `markupId`, you first must create a “dummy” viewing session, specifying the `markupId` you want to access via the `source.markupId` property of your `POST /ViewingSession` request.

However, because `source.markupId` is optional when creating a viewing session, you may not know the `markupId` to use. Fortunately, as long as you still know the information about the source document and how you created the original viewing session for your end user, there is still a way to access the markup data. When a viewing session is created without a `source.markupId` specified, we will automatically calculate a `markupId` value *based upon the other source object properties*. So, if you did not specify an explicit `source.markupId` when creating the original viewing session for your end user, you can still access the markup data that end user saved by creating a new “dummy” viewing session with exactly the same set of `source` properties. For more information about the `source` object options you can pass in on a `POST /ViewingSession` request, [see the `POST /ViewingSession` API reference](#).

Available URLs

URL	Description
GET /AnnotationList/q/Art/q	Gets the list of available annotation XML files from the server.
GET /Document/q/Art/q	Gets a specific annotations XML file from the server.
POST /Document/q/Art/q	Creates a new annotations XML file using the provided data.

GET /AnnotationList/q/Art/q?DocumentID=u{viewingSessionId}

Routes key: `GetAnnotations`

Gets the list of available annotation XML files from the server.

```
GET pas_base_url/AnnotationList/q/Art/q?DocumentID=u1234
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "annotationFiles": [
    {
      "annotationLabel": "anId",
      "annotationName": "abcd_0_anId.xml",
      "ID": "1"
    }
  ]
}
```

Error Responses

When the server's license could not be verified:

```
HTTP/1.1 480 LicenseCouldNotBeVerified
Content-Type: application/json

{
  "errorCode": "LicenseCouldNotBeVerified"
}
```

GET /Document/q/Art/q? DocumentID=u{viewingSessionId}&AnnotationID=u{annotationId}

Routes key: `GetAnnotation`

Gets a specific annotations XML file from the server.

```
GET pas_base_url/Document/q/Art/q?DocumentID=u1234&AnnotationID=uanId
```

Successful Response

```
HTTP/1.1 200 OK
Content-Type: application/xml

<?xml version="1.0"?>...
```

Error Responses

When the server's license could not be verified:

```
HTTP/1.1 480 LicenseCouldNotBeVerified
Content-Type: application/json

{
  "errorCode": "LicenseCouldNotBeVerified"
}
```

POST /Document/q/Art/q? DocumentID=u{viewingSessionId}&AnnotationID=u{annotationId}

Routes key: `CreateAnnotation`

Creates a new annotations XML file using the provided data.

```
POST pas_base_url/Document/q/Art/q?DocumentID=u1234&AnnotationID=uanotherId
Content-Type: application/xml

<?xml version="1.0">...
```

Successful Response

```
HTTP/1.1 200 OK
```

Error Responses

When the server's license could not be verified:

```
HTTP/1.1 480 LicenseCouldNotBeVerified
Content-Type: application/json

{
  "errorCode": "LicenseCouldNotBeVerified"
}
```

Search Tasks

Introduction

The *search task* REST API is used by our viewer to perform server-side searching and text retrieval against a document which is currently being viewed.

This REST API is designed primarily for our viewer. If your application needs to perform document search without a viewer involved, **we recommend you use PrizmDoc Server's [search context](#) and [search tasks](#) REST APIs directly.**

Available URLs

URL	Description
POST /v2/viewingSessions/{viewingSessionId}/searchTasks	Starts an asynchronous full-text search against a viewing session's source document.
GET /v2/searchTasks/{processId}/results	Gets available search results.
DELETE /v2/searchTasks/{processId}	Cancels a search task.

POST /v2/viewingSessions/{viewingSessionId}/searchTasks

Starts an asynchronous full-text search against a viewing session's source document.

After a successful POST to create the search task, we immediately begin a background process to start populating search results for you to [GET](#). You do not need to wait for the full set of results to be available; you can start [retrieving partial search results](#) as soon as they are available. Once the full text of the document has been searched and no more results will be added, the search task `state` will change from `"processing"` to `"complete"`.

Request

Request Headers

Name	Description
Content-Type	Must be <code>application/json</code>

Request Body

- `input`
 - `searchTerms[]` (Array of Objects) **Required and must contain at least one item.** Each item must be an object which conforms to one of the following:
 - *Simple (finds all occurrences of a single regex pattern):*
 - `type: "simple"` (String) **Required.** Must be set to `"simple"` to indicate this is a simple term object.
 - `pattern` (String) **Required.** Regular expression to search for, using a [JavaScript-flavored regular expression string](#).
 - `caseSensitive` (Boolean) Determines whether we consider case when matching this term. Default is `false`.
 - `contextPadding` (Integer) Maximum number of characters to include both before and after the search result in the returned `context` string. For example, a value of `25` would allow up to 25 preceding and 25 following characters of content. Default is `25`.
 - `termId` (String) Optional id of your choosing which, if provided, will be included as a `termId` property on each search result produced by this term. When used, we do not enforce uniqueness; it is your responsibility to use a unique `termId` for each term.
 - *Proximity (finds all occurrences of multiple regex patterns which are near each other):*
 - `type: "proximity"` (String) **Required.** Must be set to `"proximity"` to indicate this is a proximity term object.
 - `subTerms[]` (Array of Objects) **Required and must contain at least two items.** Each item may contain:
 - `pattern` (String) **Required.** Regular expression for this particular term, using a [JavaScript-flavored regular expression string](#).
 - `caseSensitive` (Boolean) Determines whether we consider case when matching this term. Default is `false`.
 - `distance` (Integer) **Required.** Maximum number of words allowed between any two consecutive sub-terms.
 - `contextPadding` (Integer) Maximum number of characters to include both before and after the search result in the returned `context` string. For example, a value of `25` would allow up to 25 preceding and 25 following characters of content. Default is `25`.
 - `termId` (String) Optional id of your choosing which, if provided, will be included as a `termId` property on each search result produced by this term. When used, we do not enforce uniqueness; it is your responsibility to use a unique `termId` for each term.
- `minSecondsAvailable` (Integer) The minimum number of seconds this search task will remain available. The actual lifetime may be longer.

Successful Response

Response Body

JSON with metadata about the created search task.

- `input` (Object) Input we accepted to create the search task.
- `processId` (String) Unique id for this search task.
- `affinityToken` (String) Affinity token for this search task. Present when clustering is enabled.
- `state` (String) State of getting search results.

- "processing" - The search is still being executed. Additional results may become available.
 - "complete" - The search is complete. No additional results will become available.
 - "error" - There was a problem performing the search. No additional results will become available.
- percentComplete (Integer) Percentage of the document text which has been searched (from 0 to 100).
- expirationDateTime (String) Currently planned date and time when the search task resource will expire and no longer be available for use. Format is [RFC 3339 Internet Date/Time profile of ISO 8601](#), e.g. "2016-11-05T08:15:30.494Z".

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No viewing session with the provided {viewingSessionId} could be found.
480	"DocumentNotProvidedYet"	The viewing session does not yet have a source document attached.
480	"MissingInput"	A required input value was not provided. See <code>errorDetails</code> in the response body.
480	"InvalidInput"	An invalid input value was used. See <code>errorDetails</code> in the response body.
480	"MissingInputForSimpleTerm"	An invalid input value was used in a "simple" term object. See <code>errorDetails</code> in the response body.
480	"InvalidInputForSimpleTerm"	An invalid input value was used in a "simple" term object. See <code>errorDetails</code> in the response body.
480	"MissingInputForProximityTerm"	An invalid input value was used in a "proximity" term object. See <code>errorDetails</code> in the response body.
480	"InvalidInputForProximityTerm"	An invalid input value was used in a "proximity" term object. See <code>errorDetails</code> in the response body.
480	"FeatureDisabled"	The viewing session was created with "serverSideSearch" disabled.
480	"ResourceNotUsable"	The underlying search resources are not usable for this viewing session.
580	"InternalError"	The server encountered an internal error when handling the request.

Example

Request

This POST begins a search task which finds all instances of the word "quick":

```
POST pas_base_url/v2/viewingSessions/XYZ.../searchTasks
Content-Type: application/json

{
```

```
"input": {
  "searchTerms": [{
    "type": "simple",
    "pattern": "quick"
  }]
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "searchTerms": [{
      "type": "simple",
      "pattern": "quick",
      "caseSensitive": false,
      "contextPadding": 25
    }]
  },
  "processId": "pR5X6nPDgMwat6cxlmn0Q3",
  "state": "processing",
  "percentComplete": 0,
  "expirationDateTime": "2016-12-17T20:38:39.796Z"
}
```

Additional Examples

Start a case-sensitive search for an exact phrase

This POST begins a case-sensitive search for the exact phrase "The quick brown fox jumped over the lazy dog.". Notice that we had to escape the period character because it is a special regex character (`\.`), and because this is a JSON string value, the backslash itself must also be escaped (`"\\."`):

```
POST pas_base_url/v2/viewingSessions/XYZ.../searchTasks
Content-Type: application/json

{
  "input": {
    "searchTerms": [{
      "type": "simple",
      "pattern": "The quick brown fox jumped over the lazy dog\\.",
      "caseSensitive": true
    }]
  }
}
```

Start a search for every instance of the word "quick" or "brown" or "fox"

This POST begins a search for the words "quick" or "brown" or "fox", locating all instances of each of these words:

```
POST pas\_base\_url/v2/viewingSessions/XYZ.../searchTasks
Content-Type: application/json
```

```
{
  "input": {
    "searchTerms": [{
      "type": "simple",
      "pattern": "quick"
    }, {
      "type": "simple",
      "pattern": "fox"
    }, {
      "type": "simple",
      "pattern": "dog"
    }]
  }
}
```

Start a search for "quick" and "fox" and "dog" where there are no more than 5 words between any two consecutive occurrences of them

```
POST pas\_base\_url/v2/viewingSessions/XYZ.../searchTasks
Content-Type: application/json
```

```
{
  "input": {
    "searchTerms": [{
      "type": "proximity",
      "subTerms": [{
        "pattern": "quick"
      }, {
        "pattern": "fox"
      }, {
        "pattern": "dog"
      }]
    }, {
      "distance": 5
    }]
  }
}
```

Start a case-sensitive search for "John Doe" within 30 words of what looks like a Social Security Number

```
POST pas\_base\_url/v2/viewingSessions/XYZ.../searchTasks
Content-Type: application/json
```

```
{
  "input": {
    "searchTerms": [{
      "type": "proximity",
      "subTerms": [{
        "pattern": "John Doe",
        "caseSensitive": true
      }, {
        "pattern": "\\d{3}-\\d{2}-\\d{4}"
      }]
    }
  ]
}
```

```

    }],
    "distance": 30
  }]
}
}

```

GET /v2/searchTasks/{processId}/results?limit={limit}&continueToken={continueToken}

Gets a block of newly-available search results up to a limit.

This URL is designed to give you the results in chunks as they become available. Each GET request will return the currently-known results up to a `limit` (default is 100). If a response contains a `continueToken`, it indicates that additional results may be available and that you should issue another GET request using that `continueToken` as a query string parameter to skip the results you have already received. As long as a response contains a `continueToken`, use it to issue a subsequent GET for more results. When you encounter a response which does *not* have a `continueToken`, you have received all of the results and no more GET requests are necessary.

In order to optimize the number of network requests you make, any response which contains a `continueToken` will also contain a `continueAfter` value with a recommended number of milliseconds you should wait before sending the next GET request.

Request

URL Parameters

Parameter	Description
{processId}	The <code>processId</code> which identifies the search task.
{limit}	The maximum number of results to return for this HTTP request. Must be an integer greater than 0. Default is 100.
{continueToken}	Used to continue getting results from the point where a previous GET request left off.

Request Headers

Name	Description
Accusoft-Affinity-Token	The <code>affinityToken</code> of the search task. Required when server clustering is enabled.

Successful Response

Response Body

JSON with any available search results.

- `results` (Array of Objects) **Always present.** Array of newly-available search results. If no new results are available, this array will be empty.
 - `id` (Integer) Unique number assigned to this search result.
 - `pageIndex` (Integer) Zero-indexed page number where this search result occurs in the document.
 - `text` (String) Text which was matched.

- `context` (String) Contextual excerpt, including the matched text itself. The amount of leading and trailing characters to include in this value is controlled by `input.contextPadding` in the initial POST to create the search task.
- `boundingRectangle` (Object) Bounding rectangle dimensions of the matched text on the page where it occurs.
 - `x` (Number) Distance from the left edge of the page to the left edge of the search result bounding box.
 - `y` (Number) Distance from the top edge of the page to the top edge of the search result bounding box.
 - `width` (Number) Width of the search result bounding box.
 - `height` (Number) Height of the search result bounding box.
- `lineRectangles` (Array of Objects) Array of rectangles for each line of the matched text on the page where it occurs. If the match is on one line, the result is a single array item with a rectangle equal to `boundingRectangle`. If the match is on multiple lines, all rectangles in the array will be within the bounds of the `boundingRectangle`.
 - `x` (Number) Distance from the left edge of the page to the left edge of the search result line rectangle.
 - `y` (Number) Distance from the top edge of the page to the top edge of the search result line rectangle.
 - `width` (Number) Width of the search result line rectangle.
 - `height` (Number) Height of the search result line rectangle.
- `pageData` (Object) Information about the dimensions of the page where this search result occurs.
 - `width` (Number) Width of the page.
 - `height` (Number) Height of the page.
- `searchTerm` (Object) Search term which produced this result. The value will correspond to one of the items passed in to `input.searchTerms` in the initial POST to create the search task.
 - *When type is "simple":*
 - `type` (String) **Always present** with a value of "simple".
 - `pattern` (String) **Always present.** Regular expression which produced the result.
 - `caseSensitive` (Boolean) **Always present.** Indicates whether or not case was considered for this result.
 - `contextPadding` (Integer) **Always present.** Amount of context padding requested for this term in the initial POST.
 - `termId` (String) When provided in the initial POST, `termId` of the term which produced this result.
 - *When type is "proximity":*
 - `type` (String) **Always present** with a value of "proximity".
 - `subTerms []` (Array of Objects) **Always present.** The sub-terms which contributed to this result. Each item will contain:
 - `pattern` (String) **Always present.** Regular expression for this particular sub-term.
 - `caseSensitive` (Boolean) **Always present.** Indicates whether or not case was considered when matching this particular sub-term in the result.
 - `distance` (Integer) **Always present.** Maximum number of words allowed between any two consecutive sub-terms.
 - `contextPadding` (Integer) **Always present.** Amount of context padding requested for this term in the initial POST.
 - `termId` (String) When provided in the initial POST, `termId` of the term which produced this result.
- `startIndex` (Integer) JavaScript string index into the full-page text string where the matched text begins.
- `startIndexInContext` (Integer) JavaScript string index into the returned `context` string where

the matched text begins.

- `pagesWithoutText` (Array of Integers) **Always present.** Currently known pages in the document which do not contain any text content at all. Values are zero-indexed page numbers. If the search task is still processing (a `continueToken` is present in the response), the data should be considered partial. Note that, unlike `results`, this value is cumulative (we always deliver the entire set of pages we know to not contain text data).
- `continueToken` (String) When present, indicates that more search results may be available. An additional GET request should be made for more results using this value as the `continueToken` query string parameter. When not present, indicates that the search is complete and no further results will be available.
- `continueAfter` (Number) Recommended milliseconds to delay before issuing the next GET request for more results.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No search task with the provided <code>{processId}</code> could be found.
480	"MissingInput"	A required input value was not provided. See <code>errorDetails</code> in the response body.
480	"InvalidInput"	An invalid input value was used. See <code>errorDetails</code> in the response body.
480	"ResourceNotUsable"	Can occur when the search task is in a state of "error".
580	"InternalError"	The server encountered an internal error when handling the request.

Example

Say you have a search task which was created to find the regex `"manag[a-z]*"` in a particular whitepaper. Here is an example sequence of requests and responses illustrating how you would acquire the full set of results for the search task (for brevity, the total number of search results in this example is small).

You would start with an initial GET:

```
GET pas_base_url/v2/searchTasks/pR5X6nPDgMwat6cxlmn0Q3/results
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "results": [
    {
      "id": 0,
      "pageIndex": 0,
      "text": "Management",
      "context": "Self-Hosted Content Management Best Practices",
      "boundingRectangle": { "x": 24.20, "y": 13.74, "width": 234.20, "height":
26.10 },
      "lineRectangles": [{ "x": 24.20, "y": 13.74, "width": 234.20, "height":
26.10 }],
      "pageData": { "width": 612, "height": 792 },
      "searchTerm": {
```

```

        "type": "simple",
        "pattern": "manag[a-z]*",
        "caseSensitive": false,
        "contextPadding": 25
    },
    "startIndex": 19,
    "startIndexInContext": 19
},
{
    "id": 1,
    "pageIndex": 0,
    "text": "management",
    "context": "ue of enterprise content management software should go way b",
    "boundingRectangle": { "x": 156.07, "y": 352.19, "width": 105.00, "height":
13.41 },
    "lineRectangles": [{ "x": 156.07, "y": 352.19, "width": 105.00, "height":
13.41 }],
    "pageData": { "width": 612, "height": 792 },
    "searchTerm": {
        "type": "simple",
        "pattern": "manag[a-z]*",
        "caseSensitive": false,
        "contextPadding": 25
    },
    "startIndex": 527,
    "startIndexInContext": 25
}
],
"pagesWithoutText": [],
"continueToken": "Cx07GH1kmi32gxAQhv49WZ",
"continueAfter": 500
}

```

The initial response has given us two results for the first page of the document (page index 0) and a `continueToken` which we should use to get more results after waiting 500 milliseconds.

So, half a second later, we issue a follow-up request with the `continueToken` passed in as a query string parameter (so we skip over the results we already have):

```

GET pas\_base\_url/v2/searchTasks/pR5X6nPDgMwat6cxlmn0Q3/results?
continueToken=Cx07GH1kmi32gxAQhv49WZ
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

```

```

HTTP/1.1 200 OK
Content-Type: application/json

```

```

{
  "results": [
    {
      "id": 2,
      "pageIndex": 1,
      "text": "management",
      "context": "Self-Hosted content management software helps eliminate",
      "boundingRectangle": { "x": 310.21, "y": 562.14, "width": 254.03, "height":
26.10 },
      "lineRectangles": [{ "x": 310.21, "y": 562.14, "width": 254.03, "height":
26.10 }],

```



```

    "pageData": { "width": 612, "height": 792 },
    "searchTerm": {
      "type": "simple",
      "pattern": "manag[a-z]*",
      "caseSensitive": false,
      "contextPadding": 25
    },
    "startIndex": 652,
    "startIndexInContext": 19
  }
],
"pagesWithoutText": [2,3],
"continueToken": "B4uGe7m0ZtxR3lkqA07Nmj",
"continueAfter": 500
}

```

This time we get back a new result as well as some new information about `pagesWithoutText`: we now know that at least page indices 2 and 3 (zero-indexed page numbers) have no text at all.

The presence of a new `continueToken` tells us there may be more results, so we submit another request with the new `continueToken`:

```

GET pas\_base\_url/v2/searchTasks/pR5X6nPDgMwat6cxlmn0Q3/results?
continueToken=B4uGe7m0ZtxR3lkqA07Nmj
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

```

```

HTTP/1.1 200 OK
Content-Type: application/json

{
  "results": [
    {
      "id": 3,
      "pageIndex": 5,
      "text": "management",
      "context": "upply chains to contract management, or HR processes to gove",
      "boundingRectangle": { "x": 67.00, "y": 142.53, "width": 254.03, "height":
26.10 },
      "lineRectangles": [{ "x": 67.00, "y": 142.53, "width": 254.03, "height":
26.10 }],
      "pageData": { "width": 612, "height": 792 },
      "searchTerm": {
        "type": "simple",
        "pattern": "manag[a-z]*",
        "caseSensitive": false,
        "contextPadding": 25
      },
      "startIndex": 113,
      "startIndexInContext": 25
    }
  ],
  "pagesWithoutText": [2,3,4]
}

```

This time we get a new result for page index 5, and we now know that page indices 2, 3, and 4 all contain no text

at all (apparently this was not much of a whitepaper!). The lack of a `continueToken` tells us we have received all of the results, so there are no more GET requests to make.

DELETE /v2/searchTasks/{processId}

Cancels a search task. Further requests using this `processId` will return errors.

Request

URL Parameters

Parameter	Description
<code>{processId}</code>	The <code>processId</code> which identifies the search task.

Request Headers

Name	Description
<code>Accusoft-Affinity-Token</code>	the <code>affinityToken</code> of the search task. Required when server clustering is enabled.

Successful Response

```
HTTP/1.1 204 No Content
```

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No search task with the provided <code>{processId}</code> could be found.
400	"MissingInput"	Can occur when clustering is enabled and an <code>Accusoft-Affinity-Token</code> was not provided.
580	"InternalError"	The server encountered an internal error when handling the request.

PrizmDoc Server REST API

Overview

The PrizmDoc Server REST API is for **automated document processing**.

For viewing functionality, use the [PAS REST API](#) instead.

For application development in .NET, we recommend using the [PrizmDoc Server .NET SDK](#) instead of using the PrizmDoc Server REST APIs directly.

General Information

- [Base URL for PrizmDoc Server](#)

Areas of Functionality

The PrizmDoc Server REST APIs can be broken into the following groups:

Application Development

These are the REST APIs you will most-commonly use:

- [Content Conversion Service](#)
- [Markup Burners](#)
- [Plain Text Redactors](#)
- [Redaction Creator](#)
- [Search Contexts](#)
- [Search Tasks](#)
- [Work Files](#)

Self-Hosted Administration

These REST APIs are useful if you are self-hosting PrizmDoc Server instances:

- [Health Status](#)
- [Cluster Management](#)

Viewing Support

These REST APIs are used by PAS and our viewer. It is uncommon for your application to need to use them:

- [Attachments](#)
- [Form Extractors](#)
- [HTML5 Viewing](#)
- [Viewing Sessions](#)

Unsupported Routes

These unsupported REST APIs are only for internal use by Accusoft products and components.

- [Unsupported Routes](#)

General Information

This section contains the following information:

- [Base URL for PrizmDoc Server](#)

Base URL for PrizmDoc Server

Introduction

When making REST API calls to PrizmDoc Server, you need to use the appropriate base URL.

PrizmDoc Cloud

If you are using our PrizmDoc Cloud offering, the base URL for the PrizmDoc Server APIs varies depending on your region:

Region	PrizmDoc Server Base URL
United States	<code>https://api.accusoft.com</code>

Remember that [PrizmDoc Cloud](#) requires you to authenticate each request with an `acs-api-key` request header.

PrizmDoc Viewer Self-Hosted (Self-Hosted)

When hosting PrizmDoc Server yourself, just use the hostname and port of your PrizmDoc Server instance (or the hostname and port of the load balancer which sits in front of your PrizmDoc Server cluster).

For a typical installation on localhost, the PrizmDoc Server base URL will be:

`http://localhost:18681`

Application Development

These are the REST APIs you will most commonly use:

- [Content Conversion Service](#)
- [Markup Burners](#)
- [Plain Text Redactors](#)
- [Redaction Creator](#)
- [Search Contexts](#)
- [Search Tasks](#)
- [Work Files](#)

Content Conversion Service

Introduction

The *content converters* REST API allows your application to convert files from a variety of input formats to several common output formats.

For application development in .NET, we recommend using the [PrizmDoc Server .NET SDK](#) instead of the REST API. See the [.NET SDK How to Guides](#) for examples of how to perform file conversion with the .NET SDK.

To convert a file using the REST API:

1. Upload a file you want to use as input using the [WorkFile API](#).
2. Start a conversion operation by using the [POST](#) URL below.
3. Check the status of the conversion by (repeatedly) using the [GET](#) URL below.
4. When complete, a separate output file will exist which you can download via the [WorkFile API](#).

Available URLs

URL	Description
POST /v2/contentConverters	Create and start a content conversion
GET /v2/contentConverters/{processId}	Get the status of a content conversion

Note that these URLs begin with `/v2`, **not** `/PCCIS/V1`.

POST /v2/contentConverters

Creates a new contentConverter resource which represents the conversion process and begins converting one or multiple input files which you have previously uploaded using the [WorkFile API](#). A successful response will include a unique `processId` which identifies this contentConverter. You will use this `processId` in subsequent [GET](#) calls to get the status and final results of the conversion.

Request Headers

Name	Value	Details
Content-Type	application/json	required
Accusoft-Affinity-Token	The <code>affinityToken</code> of the work files involved in the input to the process. Example: rcgmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w=	Required when server clustering is enabled. Providing this value is important to ensuring the process will execute on the machine where the input work files actually exist. NOTE: If you do not provide the required <code>Accusoft-Affinity-Token</code>, the POST itself will succeed but the process itself will likely fail.

Request Body

At a high level, your request body should be JSON containing an `input` object with details about the `sources` and `dest` for the conversion.

Here is a minimal example:

```
POST prizmdoc_server_base_url/v2/contentConverters
Content-Type application/json

{
  "input": {
    "sources": [
      {
        "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
      }
    ],
    "dest": {
      "format": "pdf"
    }
  }
}
```

Additional options are available. Here is the full reference:

`input.sources`

The `input.sources` object specifies an array of objects, one for each input file.

Currently multiple input files are only supported when the destination format is `pdf` or `tiff`, but a future version of the product may allow you to submit multiple input source files for other destination formats.

Name	Description	Details
<code>input.sources[n].fileId</code>	Id of the WorkFile to use as input. See Supported Input File Formats	string, required Example: "ek5Zb123oYHSUEVx1bUrVQ"
<code>input.sources[n].pages</code>	Page numbers and/or page ranges separated by commas. Currently <code>pages</code> is only supported when the destination format is <code>pdf</code> or <code>tiff</code> , and is ignored otherwise. We expect this to change in a future version, in which case <code>pages</code> will be supported for other destination formats.	string, optional Example: 1, 3, 5-10
<code>input.sources[n].password</code>	Password to open the document. Currently <code>password</code> is only supported when the source format is <code>PDF</code> , <code>MS Word</code> , <code>MS Excel</code> , <code>MS PowerPoint</code> or <code>OpenDocument</code> , and is ignored otherwise. Please note that only Office Open XML versions of <code>MS Word</code> , <code>MS Excel</code> and <code>MS PowerPoint</code> are supported when <code>fidelity.msOfficeDocumentsRenderer</code> is set to "libreoffice". We expect this to change in a future version, in which case <code>password</code> will be supported for other source formats.	string, optional Example: "secret"
<code>input.sources[n].wordOptions</code>	Additional options of a Microsoft Word compatible source file format.	object, optional
<code>input.sources[n].powerPointOptions</code>	Additional options of a Microsoft PowerPoint compatible source file format.	object, optional

`input.sources[n].wordOptions`

The `input.sources[n].wordOptions` object will be considered only when the source is a supported Microsoft Word format:

- doc
- docx
- docm
- dot
- dotx
- dotm

Otherwise it will be ignored. The object has a `trackedChanges` enumeration property that controls how tracked changes should be handled. Supported values:

- "preserve" - Preserve all tracked changes, neither accepting nor rejecting them before creating the output file. Output PDFs, images, or SVGs will visually show the tracked changes in a different color (additions as colored text and deletions as struck-through colored text).
- "acceptAll" - Accept all tracked changes before creating the output file(s).
- "rejectAll" - Reject all tracked changes before creating the output file(s).

The default is "preserve" (where output PDFs, images, and SVGs will visually show the tracked changes). If you want to create output files where tracked changes are not shown, specify one of the other options.

NOTE: In order to use "acceptAll" and "rejectAll" values, the *Microsoft Office rendering mode* should be enabled by the MSO feature in your license key.

Here is a minimal example:

```
POST prizmdoc\_server\_base\_url/v2/contentConverters
Content-Type application/json

{
  "input": {
    "sources": [
      {
        "fileId": "ek5Zb123oYHSUEVx1bUrVQ",
        "wordOptions": {
          "trackedChanges": "preserve"
        }
      },
      {
        "fileId": "ElkNzWtrUJp4rXI5YnLUgw",
        "wordOptions": {
          "trackedChanges": "acceptAll"
        }
      }
    ],
    "dest": {
      "format": "pdf"
    }
  }
}
```

input.sources[n].powerPointOptions

The `input.sources[n].powerPointOptions` object will be considered only when the source is a supported Microsoft PowerPoint format:

- ppt
- pptx
- pptm
- pot
- potx
- potm
- pps
- ppsx
- ppsm

Otherwise it will be ignored. The object has a `renderingMode` enumeration property that controls how the document should be rendered. Supported values:

- "slides" - Put only slides into the output file(s).
- "slidesWithNotes" - Put slides along with speaker notes into the output file(s).

The default is "slides" (where output PDFs, images, and SVGs will contain slides only). If you want to create output files with speaker notes shown, please specify "slidesWithNotes" option.

NOTE: In order to use "slidesWithNotes" value, the *Microsoft Office rendering mode* should be enabled by the MSO feature in your license key.

Here is a minimal example:

```
POST prizmdoc\_server\_base\_url/v2/contentConverters
Content-Type application/json

{
  "input": {
    "sources": [
      {
        "fileId": "eV3tA13awOLE7i-s1FkUA",
        "powerPointOptions": {
          "renderingMode": "slides"
        }
      },
      {
        "fileId": "YJ6jG1EM8eT7TggX5TD8og",
        "wordOptions": {
          "renderingMode": "slidesWithNotes"
        }
      }
    ],
    "dest": {
      "format": "pdf"
    }
  }
}
```

input.src (deprecated)

The `input.src` object specifies the file to use as input. This property has been deprecated, please use `input.sources` instead.

Name	Description	Details
<code>input.src.fileId</code>	Id of the WorkFile to use as input. See Supported Input File Formats	string, required Example: "ek5Zb123oYHSUEVx1bUrVQ"

input.dest

The `input.dest` object specifies the destination file format and any additional details which control how the content is converted.

Name	Description	Details
<code>input.dest.format</code>	<p>Output file format.</p> <p>Supported values:</p> <ul style="list-style-type: none"> • "jpeg" • "pdf" • "png" • "svg" • "tiff" • "docx" • "xlsx" <p>Currently "docx" destination format is only supported when the source format is PDF, and <code>input.features.pdfToDocx.enabled</code> is enabled. This functionality also requires the Microsoft Office rendering mode to be enabled by the MSO feature in your license key.</p> <p>The "xlsx" destination format is only supported when the source format is either XLS or CSV. This functionality also requires the Microsoft Office rendering mode to be enabled by the MSO feature in your license key.</p>	string, required
<code>input.dest.jpegOptions</code>	Additional options when <code>input.dest.format</code> is "jpeg".	object, optional
<code>input.dest.pdfOptions</code>	Additional options when <code>input.dest.format</code> is "pdf".	object, optional
<code>input.dest.pngOptions</code>	Additional options when <code>input.dest.format</code> is "png".	object, optional
<code>input.dest.tiffOptions</code>	Additional options when <code>input.dest.format</code> is "tiff".	object, optional
<code>input.dest.header</code>	Header to append to each page of the output. The page dimensions will be expanded to allow space for the additional header content (the original page content will be unaltered).	object, optional
<code>input.dest.footer</code>	Footer to be append to each page of the output. The page dimensions will be expanded to allow space for the additional footer content (the original page content will be unaltered).	object, optional
<code>input.dest.watermarks</code>	Watermarks to apply to each page of the output.	object, optional

input.dest.jpegOptions

Name	Description	Details
<code>input.dest.jpegOptions.maxWidth</code>	Maximum pixel width of the output JPEG image, expressed as a CSS-style string, e.g. "800px". When specified, the output will never be wider than the specified value and its aspect ratio will be preserved.	string, optional Example: "800px"
<code>input.dest.jpegOptions.maxHeight</code>	Maximum pixel height of the output JPEG image, expressed as a CSS-style string, e.g. "600px". When specified, the output will never be taller than the specified value and its aspect ratio will be preserved.	string, optional Example: "600px"

For CAD input, you must specify either `maxWidth` or `maxHeight`.

input.dest.pdfOptions

Name	Description	Details
<code>input.dest.pdfOptions.forceOneFilePerPage</code>	When <code>true</code> , forces output to be a collection of single-page PDF files (rather than a single multi-page PDF file). Default is <code>false</code> .	boolean, optional
<code>input.dest.pdfOptions.ocr</code>	Options for text recognition. Applies when the source file is raster or image-based PDF (a PDF which has a single raster image on each page). > NOTE: If you are attempting to make a searchable PDF from an existing PDF document, please note that the source PDF file should be an image-only PDF. PrizmDoc will not create a searchable file from already-existing vector content.	object, optional
<code>input.dest.pdfOptions.ocr.language</code>	Language to recognize. Currently, only English is supported. Value must be "english".	string, required
<code>input.dest.pdfOptions.ocr.defaultDpi</code>	Default DPI to use when the input does not specify DPI information. Default is { x: 300, y: 300 }	object, optional
<code>input.dest.pdfOptions.ocr.defaultDpi.x</code>	Horizontal DPI value.	integer, required
<code>input.dest.pdfOptions.ocr.defaultDpi.y</code>	Vertical DPI value.	integer, required

When converting PDF documents to a single PDF with multiple pages or a set of single-page PDF files, the result PDF file(s) will lose bookmarks and intra-document links due to restructuring of the PDF content.

The width and height of the recognized image should not exceed 32767 pixels.

Strikethrough text will not be recognized.

input.dest.pngOptions

Name	Description	Details
<code>input.dest.pngOptions.maxWidth</code>	Maximum pixel width of the output PNG image, expressed as a CSS-style string, e.g. "800px". When specified, the output will never be wider than the specified value and its aspect ratio will be preserved.	string, optional Example: "800px"
<code>input.dest.pngOptions.maxHeight</code>	Maximum pixel height of the output PNG image, expressed as a CSS-style string, e.g. "600px". When specified, the output will never be taller than the specified value and its aspect ratio will be preserved.	string, optional Example: "600px"

For CAD input, you must specify either `maxWidth` or `maxHeight`.

input.dest.tiffOptions

Name	Description	Details
<code>input.dest.tiffOptions.forceOneFilePerPage</code>	When <code>true</code> , forces output to be a collection of single-page TIFF files (rather than a single multi-page TIFF file). Default is <code>false</code> .	boolean, optional
<code>input.dest.tiffOptions.maxWidth</code>	Maximum pixel width of each page of the output TIFF, expressed as a CSS-style string, e.g. "800px". When specified, the output pages are guaranteed to never be wider than the specified value and their aspect ratio will be preserved.	string, optional Example: "800px"
<code>input.dest.tiffOptions.maxHeight</code>	Maximum pixel height of each page of the output TIFF, expressed as a CSS-style string, e.g. "600px". When specified, the output pages are guaranteed to never be taller than the specified value and their aspect ratio will be preserved.	string, optional Example: "600px"
<code>input.dest.tiffOptions.compression.type</code>	Output image compression type to use. Supported values: <ul style="list-style-type: none"> "auto" - Automatically choose output compression type based on the source document(s). "lzw" - Force the use of LZW compression. LZW is a lossless compression format which preserves transparency. "jpeg" - Force the use of JPEG compression. JPEG is a lossy compression format ideal for photographs. "jpeg" compression can only be used when <code>input.dest.tiffOptions.color.mode</code> is "auto" (the default), "grayscale", or "rgb". "group4" - Force the use of Group 4 compression. Group 4 is only for bitonal (black and white) output. As such, "group4" can only be used when <code>input.dest.tiffOptions.color.mode</code> is "auto" (the default) or "bitonal". Default is "auto"	string, optional Example: "lzw"
<code>input.dest.tiffOptions.color.mode</code>	Output image color mode to use. Supported values: <ul style="list-style-type: none"> "auto" - Automatically choose output color mode based on the source document(s). "bitonal" - Force the use of bitonal color mode, limiting the output to black and white only, 1-bit per pixel. "grayscale" - Force the use of grayscale color mode, limiting the output to grayscale colors, 8-bits per pixel. "indexed" - Force the use of indexed color mode, limiting the output to 256 colors, 8-bits per pixel. "rgb" - Force the use of RGB color, 24-bits per pixel. Default is "auto".	string, optional Example: "indexed"
<code>input.dest.tiffOptions.dpi</code>	Output image resolution (in dots per inch) to use. When specified (it should be a positive integer), the output TIFF image will have requested <code>dpi</code> value and will be scaled if necessary. Please note that <code>input.dest.tiffOptions.maxWidth</code> and <code>input.dest.tiffOptions.maxHeight</code> have precedence over <code>input.dest.tiffOptions.dpi</code> , so if either of those parameters is present, the output image will be resized according to them, and the new resolution will be assigned without additional resizing. Otherwise, if neither of <code>maxWidth</code> / <code>maxHeight</code> is specified, the <code>dpi</code> resolution will be applied as follows: <ul style="list-style-type: none"> If the source image has a valid resolution, the output image will be resized by scale factor $\text{input.dest.tiffOptions.dpi} / \text{imageCurrentResolution}$ If the source image has no valid resolution, the new resolution will be assigned without resizing 	integer, optional Example: 300

Name	Description	Details
------	-------------	---------

For CAD input, you must specify either `maxWidth` or `maxHeight`.

`input.dest.header`

Name	Description	Details
<code>input.dest.header.lines</code>	Text content for the header. This is a multi-dimensional array that allows you to easily position text left, center, and right. The first string in any inner array will always be placed on the left (left-justified), the second string placed in the center (center-justified), and the third string placed on the right (right-justified). The number of items in the outer array defines the total number of text lines. You may provide between one and three lines of text for a header.	array, optional
<code>input.dest.header.fontFamily</code>	Font family to use (e.g. "Courier"). The font name provided must be present on the server to be applied.	string, optional
<code>input.dest.header.fontSize</code>	Font size in points. Value must be a string with a number followed by "pt" (e.g. "12pt").	string, optional
<code>input.dest.header.color</code>	Text color. Value must be in 6-digit CSS hex color format (e.g. "#FF0000").	string, optional

Currently, the `input.dest.header` property is only supported when converting all pages of a single document to either "pdf" or "tiff", and `forceOneFilePerPage` is false.

Text may overlap other text and/or overflow the page bounds. The caller specifies the text position and size, and the product simply renders the text. For example, if the font size is too big, text on the left may overlap text in the center, or if the text is so long it can't fit on the page width, it may overflow the page bounds.

For `input.dest.header` code examples refer to [Conversion Input Examples](#).

`input.dest.footer`

Name	Description	Details
<code>input.dest.footer.lines</code>	Text content for the footer. This is a multi-dimensional array that allows you to easily position text left, center, and right. The first string in any inner array will always be placed on the left (left-justified), the second string placed in the center (center-justified), and the third string placed on the right (right-justified). The number of items in the outer array defines the total number of text lines. You may provide between one and three lines of text for a footer.	array, optional
<code>input.dest.footer.fontFamily</code>	Font family to use (e.g. "Courier"). The font name provided must be present on the server to be applied.	string, optional
<code>input.dest.footer.fontSize</code>	Font size in points. Value must be a string with a number followed by "pt" (e.g. "12pt").	string, optional
<code>input.dest.footer.color</code>	Text color. Value must be in 6-digit CSS hex color format (e.g. "#FF0000").	string, optional

Currently, the `input.dest.footer` property is only supported when converting all pages of a single document to either "pdf" or "tiff", and `forceOneFilePerPage` is false.

Text may overlap other text and/or overflow the page bounds. The caller specifies the text position and size, and the product simply renders the text. For example, if the font size is too big, text on the left may overlap text in the center, or if the text is so long it can't fit on the page width, it may overflow the page bounds.

For `input.dest.footer` code examples refer to [Conversion Input Examples](#).

`input.dest.watermarks`

Diagonal text watermark

Currently only diagonal text watermarks are supported. Each array item must be an object which conforms to the following:

Name	Description	Details
<code>type</code>	Must be set to "diagonalText" to indicate the object represents a diagonal text watermark.	string, required
<code>text</code>	Actual text of the watermark. To compose multiline text put new line characters (\n) in corresponding places. Within the string, you can use the following special tokens to insert dynamic values: <ul style="list-style-type: none"> <code>{{pageNumber}}</code> - Will be replaced with the current page number <code>{{pageCount}}</code> - Will be replaced with the total number of pages 	string, required
<code>opacity</code>	Opacity of the watermark. 1.0 is completely opaque, 0.0 is completely transparent. Default is 1.0.	number, optional
<code>color</code>	Text color. Can be any valid CSS color name (like "red") or hex value (like "#FF0000"). Default is "black".	string, optional
<code>fontFamily</code>	Specifies the name of the font that is used for the watermark (e.g. "fontFamily": "Courier"). The font name provided must be present on the server to be applied.	string, optional
<code>fontSize</code>	Font size in points. Value must be a string with a number followed by "pt" (e.g. "12pt").	string, optional
<code>fontWeight</code>	Font weight. <ul style="list-style-type: none"> "normal" "bold" 	string, optional

Name	Description	Details
	Default is "normal".	
fontStyle	Font style. Supported values: <ul style="list-style-type: none"> "normal" "italic" Default is "normal".	string, optional
textDecoration	Text decoration Supported values: <ul style="list-style-type: none"> "none" "underlined" Default is "none".	string, optional
slope	Controls the text angle. Supported values: <ul style="list-style-type: none"> "up" - Text will start in the lower-left corner of the page and extend upwards to the upper-right corner of the page. "down"- Text will start in the upper-left corner of the page and extend downwards to the lower-right corner of the page. Default is "up".	string, optional

Note that currently diagonal text watermarks are supported only when `input.dest.format` is "pdf".

input_features

The `input._features` object specifies a set of toggles used to enable some beta options for conversion operations.

input_features.pdfToDocx

Name	Description	Details
input._features.pdfToDocx.enabled	When <code>true</code> , enables CCS conversion from 'PDF' to 'DOCX'. This feature also requires the Microsoft Office rendering mode to be enabled by the MSO feature in your license key. Default is <code>false</code> .	boolean, required

Here is a minimal example:

```
POST prizmdoc_server_base_url/v2/contentConverters
Content-Type application/json

{
  "input": {
    "sources": [
      {
        "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
      }
    ],
    "dest": {
      "format": "docx"
    },
    "_features": {
      "pdfToDocx": { "enabled": true }
    }
  }
}
```

minSecondsAvailable

Allows you to specify a minimum number of seconds in which you can continue to GET the status of this conversion operation after the initial POST has been submitted. The default lifetime is defined by the `processIds.lifetime` [central configuration](#) parameter.

Response Body

A successful response will return JSON which contains:

- The `input` object submitted in the request, normalized to include default values.
- Information about the **status** of the conversion.

Here is an example:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "sources": [
      {
```

```

        "fileId": "ek5Zb123oYHSUEVx1bUrVQ",
        "pages": ""
    }
  ],
  "dest": {
    "format": "pdf",
    "pdfOptions": {
      "forceOneFilePerPage": false
    }
  }
},
"expirationDateTime": "2015-12-17T20:38:39.796Z",
"processId": "ElkNzWtrUJp4rXI5YnLUgw",
"state": "processing",
"percentComplete": 0
}

```

Conversion Status Details

Name	Description	Details
processId	The id of the contentConverter resource which represents the file conversion operation.	string
expirationDateTime	The date and time (in ISO 8601 Extended Format) when the contentConverter resource will be deleted.	string Example: "2015-12-17T20:38:39.796Z"
state	The current state of the conversion process, which will be one of the following: <ul style="list-style-type: none"> "processing" - The conversion is still in progress. "complete" - The conversion has completed successfully. "error" - The conversion failed due to a problem. For the initial POST, this value will almost always be "processing". Results are typically only available with a subsequent GET.	string
percentComplete	An integer from 0 to 100 that indicates what percentage of the conversion is complete.	integer Example: 0
errorCode	An error code string if a problem occurred during the conversion process.	string Example: "InvalidInput"
affinityToken	Affinity token echoed from request header. This value will only be present if PrizmDoc Server is running in cluster mode .	string Example: "rcqmuB9pAa8+4V7fhO1SXzawy/YMQU1g8lLdNDe517w="

HTTP Status Codes and Response JSON Error Codes

HTTP Status	"state" in response JSON body	"errorCode" in response JSON body	Description
200	processing	-	The contentConverter was created and the conversion process was started.
400	error	CouldNotReadRequestData	Could not read request data.
405	-	-	POST HTTP method was not used.
480	error	InvalidJson	Json error details are in errorDetails .
480	-	InvalidDimensionValue	Invalid dimension value specified for rasterization. See details in errorDetails .
480	-	InvalidInput	Invalid input. Invalid request data is referenced in the errorDetails .
480	-	InvalidPageSyntax	Invalid page specification. See errorDetails .
480	-	ForceOneFilePerPageNotSupportedWhenUsingHeaderOrFooter	forceOneFilePerPage mode is not supported when using header or footer options. Supported forceOneFilePerPage option is referenced in errorDetails .
480	-	MaxWidthOrMaxHeightMustBeSpecifiedWhenRasterizingCadInput	Max width or max height must be specified when rasterizing CAD input. See errorDetails .
480	-	MissingInput	Missing input. See errorDetails .
480	-	MultipleSourcesAreNotSupportedForThisDestinationFormat	Multiple source files or pages are not supported for this destination format.
480	-	MultipleSourceDocumentsNotSupportedWhenUsingHeaderOrFooter	Multiple source documents are not supported when using header or footer.
480	-	PagesPropertyNotSupportedWhenUsingHeaderOrFooter	Pages property is not supported for conversion with header or footer. The property is referenced in errorDetails
480	-	UnrecognizedExpression	Unrecognized expression. See errorDetails .
480	-	UnsupportedConversion	Unsupported conversion. See errorDetails .
480	-	UnsupportedDestinationFormatWhenUsingHeaderOrFooter	Unsupported destination format when using header or footer. Supported destination formats are listed in errorDetails .
480	-	UnsupportedDestinationFormatWhenUsingWatermarks	Unsupported destination format when using watermarks. Supported destination formats

HTTP Status	"state" in response JSON body	"errorCode" in response JSON body	Description
			are listed in <code>errorDetails</code> .
480	-	UnsupportedSourceFileFormat	Unsupported source file format. Unsupported file is referenced in <code>errorDetails</code> .
480	-	UnsupportedSourceFileFormatForOCR	Unsupported source file format for OCR. Unsupported file is referenced in <code>errorDetails</code> .
480	-	WorkFileDoesNotExist	Specified work file does not exist.
480	-	FeatureNotLicensed	The server's license does not allow the use of the requested feature. The unlicensed feature will be referenced by the <code>errorDetails</code> object in the response.
480	-	FeatureDisabled	Occurs when using an option that is not supported by the server's configuration (such as trying to set certain <code>wordOptions</code> or <code>powerPointOptions</code> when the server is not using Microsoft Office for rendering). See the <code>errorDetails</code> in the response for the specific input value which could not be used.
480	-	LicenseCouldNotBeVerified	The server's license could not be verified. If you are evaluating the product without a license, the product is running in evaluation mode and this particular part of the product is unavailable without a license. If you have a license, make sure you configured your license correctly, that your license has not expired, and that you have not exceeded any license limits (such as, for a Cloud License, the total number of logical CPU cores in use).
580	-	InternalServerError	Internal service error. This error can be returned for a number of different reasons. Please contact Customer Support.

GET /v2/contentConverters/{processId}

Gets the status of a content conversion operation and its final output if available.

In general, the response JSON will contain:

1. The `input` object submitted in the POST request, normalized to include default values.
2. Information about the **status** of the conversion.
3. Information about the **output** of the conversion, if available.

Requests can be sent to this URL repeatedly while the `state` is "processing".

When the `state` is "complete", the `output` section will list one or more `WorkFile` ids for each output file, and the files themselves can be downloaded using the [WorkFile API](#).

Parameters

Name	Description	Details
<code>processId</code>	The <code>processId</code> for a particular contentConverter. This <code>processId</code> was returned in the response for the initial POST.	string, required

Request Headers

Name	Value	Details
<code>Accusoft-Affinity-Token</code>	Affinity token returned in post response body for content converter specified by <code>processId</code> parameter. Example: "rcqmuB9pAa8+4V7fhO1SXzawy/YMQU1g8LdNDe5l7w="	Only used if PrizmDoc Server is running in cluster mode .

Response Body

While processing, the response will return JSON with only the processing details. For example:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "sources": [
      {
        "fileId": "ek5Zb123oYHSUEVx1bUrVQ",
        "pages": ""
      }
    ],
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    }
  },
  "expirationDateTime": "2015-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "state": "processing",
  "percentComplete": 82
}
```

Once the processing has completed, the response will return JSON showing the [WorkFile](#) id of the output file or files.

If the output format supports multiple pages (e.g. PDF or TIFF), then only a single output file will be created. For example:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "sources": [
      {
        "fileId": "ek5Zb123oYHSUEVx1bUrVQ",
        "pages": ""
      }
    ],
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    }
  },
  "expirationDateTime": "2015-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "state": "complete",
  "percentComplete": 100,
  "output": {
    "results": [
      {
        "fileId": "KORSwagsguevJ97BdmUbXi",
        "sources": [{ "fileId": "ek5Zb123oYHSUEVx1bUrVQ", "pages": "1-3" }],
        "pageCount": 3
      }
    ]
  }
}
```

If the output format does not support multiple pages (e.g. JPEG), then multiple output files will be created. For example:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "sources": [
      {
        "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
      }
    ],
    "dest": {
      "format": "jpeg"
    }
  },
  "expirationDateTime": "2015-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "state": "complete",
  "percentComplete": 100,
  "output": {
    "results": [
      {
        "fileId": "N6uDE11Ed6+JQPy0POu+8A",
        "sources": [{ "fileId": "ek5Zb123oYHSUEVx1bUrVQ", "pages": "1" }],
        "pageCount": 1
      },
      {
        "fileId": "+4b6QW90Fb9yjDak+ALFEg",
        "sources": [{ "fileId": "ek5Zb123oYHSUEVx1bUrVQ", "pages": "2" }],
        "pageCount": 1
      },
      {
        "fileId": "Lx/4z8AyJKV5eMjWKsBm5w",
        "sources": [{ "fileId": "ek5Zb123oYHSUEVx1bUrVQ", "pages": "3" }],
        "pageCount": 1
      }
    ]
  }
}
```

Conversion Status Details

Name	Description	Details
processId	The id of the contentConverter resource which represents the file conversion operation.	string
expirationDateTime	The date and time (in ISO 8601 Extended Format) when the contentConverter resource will be deleted.	string Example: "2015-12-17T20:38:39.796Z"
state	The current state of the conversion process, which will be one of the following:	string

Name	Description	Details
	<ul style="list-style-type: none"> "processing" - The conversion is still in progress. "complete" - The conversion has completed successfully. "error" - The conversion failed due to a problem. 	
percentComplete	An integer from 0 to 100 that indicates what percentage of the conversion is complete.	integer Example: 0
errorCode	An error code string if a problem occurred during the conversion process.	string Example: "CouldNotConvertFile"
affinityToken	Affinity token echoed from request header. This value will only be present if PrizmDoc Server is running in cluster mode .	string Example: "rcqmuB9pAa8+4V7fhO1SXzawy/YMQU1g8LdNDe5I7w="

Conversion Output Details

Name	Description	Details
output.results	An array of objects, one for each output file created.	object
output.results[n].fileId	The WorkFile id for an output file. Use this id to download the output file using the WorkFile API .	string
output.results[n].pageCount	The total number of pages in the output file.	integer
output.results[n].sources	An array of objects, one for each source file which contributed to this output file.	array
output.results[n].sources[n].fileId	The WorkFile id of the source input file.	string
output.results[n].sources[n].pages	The page or pages used from the source file. This will be a string value using one-based indexing. For example, if the output file represents page 2 of the source document, <code>pages</code> would have a value of "2". If the output file represents all 20 pages of a source document, <code>pages</code> would have a value of "1-20".	string Examples: "1-3" or "2"
output.results[n].src (deprecated)	An array with a single object which corresponds to <code>input.src</code> . This will only appear in the output if you used the deprecated <code>input.src</code> property instead of the new <code>input.sources</code> in the original POST request.	array

HTTP Status Codes and Response JSON Error Codes

HTTP status	"state" in response JSON body	"errorCode" in response JSON body	Additional "errorCode" location in response JSON body	Additional "errorCode" in response JSON body	Description
200	processing	-	-	-	The contentConverter was created and the conversion process was started.
200	complete	-	-	-	The conversion process was completed.
200	complete	-	output.results[n].errorCode	NoSuchPage	No such page. Problem fileId and page number are listed in <code>output.results[n].sources[0].fileId</code> , <code>output.results[n].sources[0].page</code>
200	error	CouldNotConvert	output.results[n].errorCode	CouldNotConvertFile	Could not convert file. Problem fileId is listed in <code>output.results[n].sources[0].fileId</code>
200	error	CouldNotConvert	output.results[n].errorCode	CouldNotConvertPage	Could not convert page. Problem fileId and page number are listed in <code>output.results[n].sources[0].fileId</code> , <code>output.results[n].sources[0].page</code>
200	error	CouldNotConvert	output.results[n].errorCode	InvalidPassword	Password is incorrect or missing. Problem fileId, page number and password if it was passed are listed in <code>output.results[n].sources[0].fileId</code> , <code>output.results[n].sources[0].page</code> , <code>output.results[n].sources[0].password</code>
200	error	CouldNotConvert	output.results[0].errorCode	RequestedHeaderOrFooterFontIsNotAvailable	Requested header or footer font is not available. Name of the font which is not available is listed in <code>input.dest.header.fontFamily</code> or <code>input.dest.footer.fontFamily</code>
200	error	CouldNotConvert	output.results[0].errorCode	RequestedWatermarkFontIsNotAvailable	Requested watermark font is not available. Name of the font which is not available is listed in <code>input.dest.watermarks[n].fontFamily</code>
200	error	CouldNotConvertAllFilesOrPages	output.results[n].errorCode	One or more occurrences of either of the following codes: CouldNotConvertFile, CouldNotConvertPage, NoSuchPage, InvalidPassword	Could not convert all files or pages. For CouldNotConvertFile error, problem fileId is listed in <code>output.results[n].sources[0].fileId</code> . For CouldNotConvertPage, InvalidPassword and

HTTP status	"state" in response JSON body	"errorCode" in response JSON body	Additional "errorCode" location in response JSON body	Additional "errorCode" in response JSON body	Description
					NoSuchPage errors, problem field and pageNumber are listed in <code>output.results[n].sources[0].fileId</code> , <code>output.results[n].sources[0].page</code>
404	-	ContentConverterDoesNotExist	-	-	Content converter does not exist. Invalid processId was specified in the request.
405	-	-	-	-	POST HTTP method was not used.
580	-	InternalServerError	-	-	Internal service error. This error can be returned for a number of different reasons. Please contact support.

Appendix

Supported Input File Formats

For a complete list of image and document source types supported by CCS, please refer to: [File Formats Reference](#).

Supported Output File Formats

- PDF
- TIFF
- PNG
- JPEG
- SVG
- DOCX
- XLSX

Note that the conversion from the source PDF to the output PDF file is implemented to remove all JavaScript during the source file processing. After the source file processing is complete, there is no JavaScript in the output PDF file(s).

Markup Burners

Introduction

The *markup burners* REST API allows your application to "burn" markup into a document, producing a new PDF with annotations applied.

For application development in .NET, we recommend using the [PrizmDoc Server .NET SDK](#) instead of the REST API. See the [.NET SDK How to Guides](#) for examples of how to perform markup burning and redaction with the .NET SDK.

Available URLs

URL	Description
POST /PCCIS/V1/MarkupBurner	Creates and starts a new MarkupBurner.
GET /PCCIS/V1/MarkupBurner/{processId}	Gets the status and result of an existing MarkupBurner.
POST /PCCIS/V1/ViewingSession/u{viewingSessionId}/MarkupBurner	Starts a new MarkupBurner using the source document of a viewing session and provided markup data as input.
GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/MarkupBurner/{processId}	Gets the status of a MarkupBurner for a viewing session.
GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/MarkupBurner/{processId}/Document	Gets the output result of a MarkupBurner process for a viewing session.

POST /PCCIS/V1/MarkupBurner

Creates and starts a new MarkupBurner.

NOTE: CAD files are not supported.

NOTE: When applying redactions, PDF annotations are never removed.

NOTE: There are limitations on what content is removed under the rectangle redaction box on vector content in PDF files when the markup burner burns the redactions.

- When redacting a vector line, a small portion of the lines under the redaction box may still exist under the redaction.
- When redacting a vector graphic with color filling, such as a colored in half circle, the color will still remain under the redaction rectangle.
- When redacting a vector graphic rectangle, they are fully removed only if the PDF graphic rectangle is fully overlapped by the redaction rectangle. If they are partially overlapped then the part not overlapped remains unredacted.

A MarkupBurner represents a process that runs on the server to "burn" markup into a document. The "burning" process makes the markup definitions a permanent part of a document. The server process is started by this request then a response is sent. Use the [GET /PCCIS/V1/MarkupBurner/{processId}](#) URL below to get the status and results of an in-progress or completed MarkupBurner process.

The input required to create a MarkupBurner is two WorkFile objects; one representing the JSON or XML which defines the markup to burn, the other representing the source document on which to burn the markup. Refer to the [work file API](#) topic for more information.

A new document is created that contains the burned-in markup. The source document will not be modified. The new document will be made available by a new WorkFile ID.

By default, MarkupBurner objects will be automatically deleted 20 minutes after they are created.

Request

Request Headers

Name	Description
Content-Type	Must be <code>application/json</code>
Accusoft-Affinity-Token	The <code>affinityToken</code> of the work files specified by <code>input.documentFileId</code> and <code>input.markupFileId</code> . Required when server clustering is enabled.

Request Body

In the request body, provide JSON containing the following properties:

- `input` (Object) Input to create the MarkupBurner.
 - `documentFileId` (String) **Required.** The ID of the WorkFile that represents the document to burn in the markup. This document will not be modified.
 - `redactionOptions` (Object) Redaction options. May contain:

- mode (String)** How redactions should be applied. May be one of the following:
 - "normal" - Actually redact the document, removing document content covered by redactions, drawing opaque redaction rectangles, and drawing any associated redaction reason text in the center of the rectangles.
 - "draft" - Do NOT actually redact the document. Instead, indicate which parts of the document would be redacted by drawing partially transparent redaction rectangles over the parts of the document that would be redacted. In order to avoid interfering with the original document content, redaction reason text will not be drawn in the center of the transparent redaction rectangles.
 Default is "normal".
- draftOptions (Object)** Options to apply when mode is set to "draft" (ignored otherwise). May contain:
 - opacity (number)** Controls the opacity of redactions when `input.redactionOptions.mode` is set to "draft". Must be a value between 0 and 1 where 0 is fully transparent and 1 is fully opaque. Default is 0.2.
- includeCategories (Array of String)** When present, limits the burning to only include the specified categories of mark types. The following categories are available:
 - "annotations"
 - "redactions"
 - "signatures"
 For example, if you only want to apply redactions you could specify:


```
"includeCategories": ["redactions"]
```

 Or, if you only wanted to apply annotations and signatures you could specify:


```
"includeCategories": ["annotations", "signatures"]
```
- markupFileId (String) Required.** The ID of the WorkFile that represents the JSON or XML document which contains the markup definition.
- minSecondsAvailable (Integer)** The minimum number of seconds which this MarkupBurner must remain available. The default lifetime is defined by the `processIds.lifetime` [central configuration](#) parameter. This value is ignored if it is shorter than the default value.

Successful Response

Response Body

If successful, this method returns JSON containing the following properties:

- input (Object)** Input we accepted to create the MarkupBurner.
 - documentFileId (String)** The ID of the WorkFile that represents the document to burn in the markup. This document will not be modified. This is an echo of what was passed in by the request.
 - markupFileId (String)** The ID of the WorkFile that represents the JSON or XML document which contains the markup definition. This is an echo of what was passed in by the request.
- expirationDateTime (String)** The date and time (in ISO 8601 Extended Format) when the MarkupBurner will be deleted.
- processId (String)** The ID of the MarkupBurner.
- state (String)** The current state of the markup burning process running on the server. This will always be "processing" in this response.
- percentComplete (Integer)** The percentage (0 – 100) complete of the markup burning process. This will always be 0 in this response.
- errorCode (String)** An error code string if a problem occurred during the markup burning process. This will always be null in this response.
- output (Object)**
 - documentFileId (String)** The ID of the new WorkFile that represents a new document with markup burned into it. This will always be null in this response.
- affinityToken (String)** Affinity token for this search context. Present when clustering is enabled.

Error Responses

Status Code	Description
400	Bad Request, if <code>input.documentFileId</code> , <code>input.markupFileId</code> is missing or invalid format, or if <code>minSecondsAvailable</code> is not a number.
405	Method Not Allowed, if POST HTTP method is not used.
480	with JSON <code>errorCode</code> "licenseCouldNotBeVerified". The server's license could not be verified. If you are evaluating the product without a license, the product is running in evaluation mode and this particular part of the product is unavailable without a license. If you have a license, make sure you configured your license correctly, that your license has not expired, and that you have not exceeded any license limits (such as, for a Cloud License, the total number of logical CPU cores in use).

Examples

Creating a MarkupBurner

Request

```
POST prizmdoc_server_base_url/PCCIS/V1/MarkupBurner
Content-Type: application/json
Accusoft-Affinity-Token: eJN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

{
  "input": {
    "documentFileId": "ek52b123oYHSUEVx1bUrVQ",
    "markupFileId": "aQ1BdViqmUisBuevJKO9SsW"
  },
  "minSecondsAvailable": 60
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentFileId": "ek52b123oYHSUEVx1bUrVQ",
    "markupFileId": "aQ1BdViqmUisBuevJKO9SsW"
  },
  "expirationDateTime": "2014-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "eJN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "processing",
  "percentComplete": 0,
  "errorCode": null,
  "output": null
}
```

GET /PCCIS/V1/MarkupBurner/{processId}

Gets the status and result of an existing MarkupBurner.

Requests can be sent to this URL repeatedly while `state` is "processing".

When `state` is "complete", the new document with burned-in annotations will be made available by a new WorkFile ID in the `output.documentFileId`. Refer to the [work file API](#) topic to find out how to download a WorkFile.

If the markup burning process encountered an error, the `state` property will be "error", the `errorCode` property will contain an error code string and `output` will be null.

Request

Request Headers

Name	Description
Content-Type	Must be <code>application/json</code>
Accusoft-Affinity-Token	The <code>affinityToken</code> of the work files specified by <code>input.documentFileId</code> and <code>input.markupFileId</code> . Required when server clustering is enabled.

Name	Description
------	-------------

URL Parameters

Parameter	Description
{processId}	The id of the process.

Response Body

If successful, this method returns JSON containing the following properties:

- **input** (Object) Input we accepted to create the MarkupBurner.
 - **documentFileId** (String) The ID of the WorkFile that represents the document to burn in the markup. This document will not be modified.
 - **markupFileId** (String) The ID of the WorkFile that represents the XML document which contains the markup definition.
- **expirationDateTime** (String) The date and time (in ISO 8601 Extended Format) when the MarkupBurner will be deleted.
- **processId** (String) The ID of the MarkupBurner.
- **state** (String) The current state of the markup burning process running on the server. The following values are allowed:
 - "processing" - The markup burning is in progress.
 - "complete" - The markup burning is completed.
 - "error" - The markup burning returns an error.
- **percentComplete** (Integer) The percentage (0 – 100) complete of the markup burning process.
- **errorCode** (String) An error code string if a problem occurred during the markup burning process.
- **output** (Object)
 - **documentFileId** (String) The ID of the new WorkFile that represents a new document with markup burned into it.
- **affinityToken** (String) Affinity token for this search context. Present when clustering is enabled.

Error Responses

Status Code	Description
404	Not Found, if {processId} does not exist.
405	Method Not Allowed, if GET HTTP method is not used.

Examples

Request

```
GET /prizmdoc_server_base_url/FCCIS/V1/MarkupBurner/ElkNzWtrUJp4rXI5YnLUgw
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

Response when the state is "processing"

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentFileId": "ek52b123oYHSUEVx1bUrVQ",
    "markupFileId": "aQ1BdViqmUisBuevJKO9Sv"
  },
  "expirationDateTime": "2014-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "processing",
  "percentComplete": 100,
  "errorCode": null,
  "output": null
}
```

Response when the state is "complete"

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentFileId": "ek52b123oYHSUEVx1bUrVQ",
    "markupFileId": "aQ1BdViqmUisBuevJKO9Sv"
  },
  "expirationDateTime": "2014-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "complete",
  "percentComplete": 100,
  "errorCode": null,
  "output": {
    "documentFileId": "vry3FPE0zQqWwhzndRccOQ"
  }
}
```

Response when the state is "error" because the work file that represents document to burn could not be found

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentFileId": "ek52b123oYHSUEVx1bUrVQ",
    "markupFileId": "aQ1BdViqmUisBuevJKO9Sv"
  },
  "expirationDateTime": "2014-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "error",
  "percentComplete": 0,
  "errorCode": "DocumentFileIdDoesNotExist"
}
```

Response when the state is "error" because the work file that contains markup could not be found

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentFileId": "ek52b123oYHSUEVx1bUrVQ",
    "markupFileId": "aQ1BdViqmUisBuevJKO9Sv"
  },
}
```

```

"expirationDateTime": "2014-12-17T20:38:39.796Z",
"processId": "ElkNzWtrUJp4rXI5YnLUgw",
"affinityToken": "ejN9/kXEYOuken4Pb91c9hqJK45XIad9LQNgCgQ+BkM=",
"state": "error",
"percentComplete": 0,
"errorCode": "MarkupFileIdDoesNotExist"
}
    
```

Response when the state is "error" because the markup work file does not contain valid XML markup and the extension of the markup work file is not JSON

```

HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentFileId": "ek5Zb123oYHSUEVx1bUrVQ",
    "markupFileId": "aQ1BdViqmUisBuevJKO9Ssw"
  },
  "expirationDateTime": "2014-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb91c9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "error",
  "percentComplete": 0,
  "errorCode": "RedactionError"
}
    
```

Response when the state is "error" because the work file that contains markup JSON cannot be parsed as JSON

```

HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentFileId": "ek5Zb123oYHSUEVx1bUrVQ",
    "markupFileId": "aQ1BdViqmUisBuevJKO9Ssw"
  },
  "expirationDateTime": "2014-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb91c9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "error",
  "percentComplete": 0,
  "errorCode": "InvalidJson"
}
    
```

Response when the state is "error" because the work file that contains markup JSON does not match the JSON Marks Schema

```

HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentFileId": "ek5Zb123oYHSUEVx1bUrVQ",
    "markupFileId": "aQ1BdViqmUisBuevJKO9Ssw"
  },
  "expirationDateTime": "2014-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb91c9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "error",
  "percentComplete": 0,
  "errorCode": "InvalidMarkup"
}
    
```

POST /PCCIS/V1/ViewingSession/u{viewingSessionId}/MarkupBurner

Starts a new MarkupBurner using the source document of a viewing session and a provided markup data as input. When the asynchronous process is ultimately finished, the output will be a new document which includes the provided markup as part of the document itself (the original source document of the viewing session is left unaltered).

This is a specialized URL which allows you to do markup burning against the source file of an existing viewing session without needing to use the work file API.

This request merely begins the markup burning process. Once started, you poll the status of the process using the GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/MarkupBurner/{processId} URL below to know when the process has completed.

Request

Request Headers

Name	Description
Content-Type	Specifies the type of content being provided to the markup burner process. It must be <code>application/xml</code> or <code>application/json</code> depending on the markup format used in the request body.

URL Parameters

Parameter	Description
{viewingSessionId}	The id provided in the response from POST /PCCIS/V1/ViewingSession.

Query String Parameters

Parameter	Description
{redactionMode}	How redactions should be applied. May be one of the following: <ul style="list-style-type: none"> "normal" - Actually redact the document, removing document content covered by redactions, drawing opaque redaction rectangles, and drawing any associated redaction reason text in the center of the rectangles. "draft" - Do NOT actually redact the document. Instead, indicate which parts of the document would be redacted by drawing partially transparent redaction rectangles over the parts of the document that would be redacted. In order to avoid interfering with the original document content, redaction reason text will not be drawn in the center of the transparent redaction rectangles. Default is "normal".
{redactionDraftOpacity}	Controls the opacity of redactions when <code>redactionMode</code> is set to "draft". Must be a value within 0 and 1 where 0 is fully transparent and 1 is fully opaque. Default is 0.2.
{RemoveFormFields}	Optional parameter indicating which interactive form fields to remove from the source document upon markup burner process. Currently only "acroform" value is supported. Please see example below.

Request Body

The JSON or XML markup to burn into the source document.

Response Body

If successful, a JSON object which may contain:

- `processId` (String) The id of the process.
- `state` (String) The current state of the markup burning process running on the server. This will always be "processing" in the initial POST response.

- `percentComplete` (Integer) The percentage (0 – 100) complete of the process. This will always be 0 in the initial POST response.
- `input` (null) Legacy property which exists only for backwards compatibility. Value will always be null.
- `output` (null) Legacy property which exists only for backwards compatibility. Value will always be null.

Error Responses

Status Code	Description
404	Not Found, if <code>{viewingSessionId}</code> does not exist.
405	Method Not Allowed, if POST HTTP method is not used.
480	An invalid input value was used. See <code>errorDetails</code> in the response body.

Examples

Request to burn a rectangle annotation using JSON markup

```
POST prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uDLbVh9eTmXJAmD1GeXbS9Gn3WHxs8oib2xPsW2xEFjnIDdoJcudPtxciodSYFQq6zYGabQ_rJiecdBkImTTkSA/MarkupBurner
Content-Type: application/json
{
  "marks": [
    {
      "uid": "Z2diOV8yMDE3LTazLTMxVDA3OjQ5OjExLjUyVWpFY2VnNjZy",
      "interactionMode": "Full",
      "pageNumber": 1,
      "type": "RectangleAnnotation",
      "creationDateTime": "2017-03-31T07:49:11.525Z",
      "modificationDateTime": "2017-03-31T07:49:11.526Z",
      "data": {},
      "conversation": {},
      "rectangle": {
        "x": 0,
        "y": 0,
        "width": 0,
        "height": 0
      },
      "pageData": {
        "width": 612,
        "height": 792
      },
      "borderColor": "#000000",
      "borderThickness": 4,
      "fillColor": "#FB0404",
      "opacity": 255
    }
  ]
}
```

Request to burn a rectangle annotation using JSON markup with removal of acroform fields

```
POST prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uDLbVh9eTmXJAmD1GeXbS9Gn3WHxs8oib2xPsW2xEFjnIDdoJcudPtxciodSYFQq6zYGabQ_rJiecdBkImTTkSA/MarkupBurner?RemoveFormFields=acroform
Content-Type: application/json
{
  "marks": [
    {
      "uid": "Z2diOV8yMDE3LTazLTMxVDA3OjQ5OjExLjUyVWpFY2VnNjZy",
      "interactionMode": "Full",
      "pageNumber": 1,
      "type": "RectangleAnnotation",
      "creationDateTime": "2017-03-31T07:49:11.525Z",
      "modificationDateTime": "2017-03-31T07:49:11.526Z",
      "data": {},
      "conversation": {},
      "rectangle": {
        "x": 0,
        "y": 0,
        "width": 0,
        "height": 0
      },
      "pageData": {
        "width": 612,
        "height": 792
      },
      "borderColor": "#000000",
      "borderThickness": 4,
      "fillColor": "#FB0404",
      "opacity": 255
    }
  ]
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "input": null,
  "processId": "ElkNzWtrUJp4rXISYnLUgw",
  "state": "processing",
  "percentComplete": 0,
  "errorCode": null,
  "output": null
}
```

GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/MarkupBurner/{processId}

Gets the status of a MarkupBurner for a viewing session.

Requests are typically sent to this URL repeatedly as long as the state is "processing".

When state is "complete", a new document with the provided markup burned into it will be available at:

GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/MarkupBurner/{processId}/Document

If an error occurred and the output could not be created, the state property will be "error" and the errorCode property will contain an error code string.

Request

URL Parameters

Parameter	Description
<code>{viewingSessionId}</code>	The id provided in the response from POST /ViewingSession.
<code>{processId}</code>	The id of the process.

Response Body

If successful, a JSON object which may contain:

- `processId` (String) The id of the process.
- `state` (String) The current state of the process. The following values are allowed:
 - "processing" - The markup burning is in progress.
 - "complete" - The markup burning is completed.
 - "error" - The markup burning returns an error.
- `percentComplete` (Integer) The percentage (0 – 100) complete of the process.
- `errorCode` (String) An error code string if a problem occurred during processing.
- `input` (null) Legacy property which exists only for backwards compatibility. Value will always be null.
- `output` (null) Legacy property which exists only for backwards compatibility. Value will always be null.

Error Responses

Status Code	Description
404	Not Found, if either the <code>{viewingSessionId}</code> or <code>{processId}</code> do not exist.
405	Method Not Allowed, if GET method is not used.

Examples

Request

```
GET /prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uDLbVh9sTmXJAmDlGeXbS9Gn3WHxs8oib2xPsW2xEFjnIDdoJcudPtxciodSYFQq6zYGabQ_rJIecdbkImTTkSA/MarkupBurner/5rGUUh3Qxhf6VXm8RkBPFA
```

Response when the state is "processing"

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": null,
  "processId": "5rGUUh3Qxhf6VXm8RkBPFA",
  "state": "processing",
  "percentComplete": 0,
  "errorCode": null,
  "output": null
}
```

Response when the state is "complete"

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": null,
  "processId": "5rGUUh3Qxhf6VXm8RkBPFA",
  "state": "complete",
  "percentComplete": 100,
  "errorCode": null,
  "output": null
}
```

Response when the state is "error" because the provided markup XML did not contain valid XML markup or the provided markup JSON was auto-detected as a non-text format

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": null,
  "processId": "53LCkvO8-rsDIAw95WgoFA",
  "state": "error",
  "percentComplete": 0,
  "errorCode": "RedactionError",
  "output": null
}
```

Response when the state is "error" because the provided markup JSON cannot be parsed as JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": null,
  "processId": "53LCkvO8-rsDIAw95WgoFA",
  "state": "error",
  "percentComplete": 0,
  "errorCode": "InvalidJson",
  "output": null
}
```

Response when the state is "error" because the provided markup JSON does not match the JSON Marks Schema

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": null,
  "processId": "53LCkvO8-rsDIAw95WgoFA",
  "state": "error",
  "percentComplete": 0,
  "errorCode": "InvalidMarkup",
  "output": null
}
```

GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/MarkupBurner/{processId}/Document?ContentDispositionFilename={ContentDispositionFilename}

Gets the output result of a MarkupBurner process for a viewing session.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The id provided in the response from POST /ViewingSession.
{processId}	The id of the process which identifies the MarkupBurner task as a string.
{ContentDispositionFilename}	The filename as a URL-encoded string, without extension, to be used in the Content-Disposition response header (the file extension will be appended automatically). The default value is "document".

Response Headers

Name	Description
Content-Disposition	Specifies 'attachment' disposition, RFC-2183 compatible filename parameter and an RFC-8187 compatible filename* parameter, allowing the use of non-ASCII filenames.
Content-Type	Will be application/pdf.

Response Body

The raw bytes of the PDF document with markup burned into it.

Error Responses

Status Code	Description
404	Not Found, which may occur if any of the following are true: the MarkupBurner has not completed yet or no such MarkupBurner exists or no such ViewingSession exists
405	Method Not Allowed, if GET method is not used.

Examples**Request**

```
GET
prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uDLbVh9sTmXJAmDlGeXbS9Gn3WHxs8oib2xPsW2xEFjnIDdoJcudPtxciodSYFQq6zYGabQ_rJIecdbkImTTkSA/MarkupBurner/ElkNzWtrUJp4rXI5YnLUgw/Document
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/pdf
Content-Disposition: attachment; filename="Greek____.pdf"; filename*=UTF-8''Greek%CE%91%CE%92%CE%93%CE%94.pdf

<<PDF data>>
```

Plain Text Redactors

Introduction

The *plain text redactors* REST API allows your application to take a source document and a redaction markup file as input and produce redacted plain text as output.

For application development in .NET, we recommend using the [PrizmDoc Server .NET SDK](#) instead of the REST API. See the [How to Create Redacted Plain Text](#) topic in the .NET SDK documentation for an example of how to do perform plain text redaction with the .NET SDK.

Available URLs

URL	Description
POST /v2/plainTextRedactors	Creates and starts a new plain text redactor process.
GET /v2/plainTextRedactors/{processId}	Gets the state of an existing plain text redactor process.

POST /v2/plainTextRedactors

Creates and starts a new plain text redactor process.

A plain text redactor represents a process that runs on the server and creates a plain text file with the text from the source document. Each redacted text fragment is replaced with the string "`<Text Redacted>`".

The server process to create the redaction markup is started by this request, but a response is sent before the process is complete. Use the [GET /v2/plainTextRedactors/{processId}](#) URL below to get the state and results of an in-progress or completed plain text redactor process.

A new plain text document will be created that will contain the redacted document text. The new document will be made available by a new WorkFile ID.

NOTES:

1. Plain text redactor supports the following mark types (see [Markup JSON Specification](#)):
 - TextSelectionRedaction
 - RectangleRedaction
 Other types of marks will have no effect on the result.
2. Plain text redactor will produce an empty plain text document for raster documents.
3. CAD files are not supported.

Request

Request Headers

Name	Description
Content-Type	Must be <code>application/json</code>
Accusoft-Affinity-Token	The <code>affinityToken</code> of work files specified by <code>input.source.fileId</code> and <code>input.markup.fileId</code> . Required when server clustering is enabled.

Request Body

JSON object conforming to the following:

- `input`
 - `source` (Object) **Required.** Describes the document from which text will be redacted. This document will not be modified.
 - `fileId` (String) **Required.** The ID of the WorkFile which represents the source document that will have redactions applied.
 - `markup` (Object) **Required.** Describes the JSON document which contains the markup definition.
 - `fileId` (String) **Required.** The ID of the WorkFile that represents the JSON markup document. The document must satisfy the [Markup JSON Specification](#).
 - `dest` (Object) **Required.** Describes the plain text output document.
 - `lineEndings` (String) **Required.** Describes the line endings in the plain text output document. The following values are allowed:
 - `"\r\n"` - Represents Windows-style line endings as CR followed by LF.
 - `"\n"` - Represents Unix-style line endings as LF alone.
- `minSecondsAvailable` (Integer) The minimum number of seconds this plain text redactor status will remain available via a GET request. The actual lifetime may be longer. The default lifetime is defined by the `processIds.lifetime` [central configuration](#) parameter.

Successful Response

Response body

JSON object with the following properties:

- `input` (Object) A copy of the request body `input` object.
- `processId` (String) The id of the new plain text redactor process.
- `expirationDateTime` (String) The date and time (in ISO 8601 Extended Format) when the plain text redactor will be deleted.
- `state` (String) The current state of the plain text redaction process running on the server. The response will always be "processing".
- `percentComplete` (Integer) The percentage (0 – 100) complete of the plain text redactor process. The response will always be 0.
- `affinityToken` (String) Affinity token for this plain text redactor. Present when clustering is enabled.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
480	"MissingInput"	A required input value was not provided. See <code>errorDetails</code> in the response body.
480	"InvalidInput"	An invalid input value was used. See <code>errorDetails</code> in the response body.
480	"UnrecognizedInput"	An unrecognized JSON property was provided. See <code>errorDetails</code> in the response body.
480	"ResourceNotFound"	Source document or markup work file does not exist. See <code>errorDetails</code> in the response body.
480	"LicenseCouldNotBeVerified"	The server's license could not be verified. If you are evaluating the product without a license, the product is running in evaluation mode and this particular part of the product is unavailable without a license. If you have a license, make sure you configured your license correctly, that your license has not expired, and that you have not exceeded any license limits (such as, for a Cloud License, the total number of logical CPU cores in use).
580	"InternalError"	The server encountered an internal error when handling the request.

Example

Request

```
POST /v2/plainTextRedactors
Content-Type: application/json
{
  "input": {
    "source": {
      "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
    },
    "markup": {
      "fileId": "vry3FPE0zQqYwhzndRccOQ"
    },
    "dest": {
      "lineEndings": "\n"
    }
  },
}
```

```
"minSecondsAvailable": 60
}
```

Response

```
HTTP 200 OK
Content-Type: application/json
{
  "processId": "iFhSZRvrz2vtFjcTSi9Qlg",
  "expirationDateTime": "2019-02-07T13:24:35.395Z",
  "input": {
    "source": {
      "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
    },
    "markup": {
      "fileId": "vry3FPE0zQqYwhzndRccOQ"
    },
    "dest": {
      "lineEndings": "\n"
    }
  },
  "state": "processing",
  "percentComplete": 0
}
```

GET /v2/plainTextRedactors/{processId}

Gets the state and result of an existing plain text redactor process created by [POST /v2/plainTextRedactors](#).

Requests can be sent to this URL repeatedly while `state` is "processing".

When `state` is "complete", the new plain text document containing the redacted document text will be made available by a new WorkFile ID in the `output.fileId`. See the [work file](#) API topic to find out how to download a WorkFile.

If the plain text redactor process encountered an error, the `state` property will be "error" and the `errorCode` property will contain an error code string.

Request

Request Headers

Name	Description
Accusoft-Affinity-Token	The <code>affinityToken</code> of the plain text redactor process. Required when server clustering is enabled.
Parameter	Description
{processId}	The id of the plain text redactor process.

Response body

JSON object with the following properties:

- `input` (Object) A copy of the [POST /v2/plainTextRedactors](#) request body `input` object we accepted to create the plain text redactor process.
- `processId` (String) The id of the plain text redactor process.
- `expirationDateTime` (String) The date and time (in ISO 8601 Extended Format) when the plain text redactor will be deleted.
- `state` (String) The current state of the plain text redaction process running on the server. The following values are allowed:
 - `"processing"` - The plain text redaction is in progress.
 - `"complete"` - The plain text redaction is completed.
 - `"error"` - The plain text redaction returns an error.
- `percentComplete` (Integer) The percentage (0 – 100) complete of the plain text redaction process. The percentage will be 100 when `state` is `"complete"`.
- `errorCode` (String) An error code string if a problem occurred during the plain text redaction process. Only present when `state` is `"error"`.
- `output` (Object) Describes the created plain text document. Only present when `state` is `"complete"`.
 - `fileId` (String) The ID of the new WorkFile that represents a new plain text document with redacted source document text.
- `affinityToken` (String) Affinity token for this plain text redactor. Present when clustering is enabled.

Error Responses and JSON Error Codes

Status Code	JSON <code>errorCode</code>	Description
404	<code>"ResourceNotFound"</code>	Text redactor process specified by <code>{processId}</code> could be found.
200	<code>"InvalidJson"</code>	Provided markup work file cannot be parsed as JSON.
200	<code>"InvalidMarkup"</code>	Provided markup JSON does not satisfy the Markup JSON Specification .
200	<code>"MarkRefersToNonExistentPage"</code>	Provided markup JSON refers to a non-existent page in the source document. See <code>errorDetails</code> in the response body.
200	<code>"UnsupportedSourceDocument"</code>	Provided source document is a CAD file and it is not supported.
580	<code>"InternalError"</code>	The server encountered an internal error when handling the request.

Example

Request

```
GET /v2/plainTextRedactors/iFhSZRvrz2vtFjcTSi9Qlg
```

Successful Response

When the plain text redactor process completes with no errors:

```
HTTP 200 OK
Content-Type: application/json
```

```
{
  "input": {
    "source": {
      "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
    },
    "markup": {
      "fileId": "vry3FPE0zQqYwhzndRccOQ"
    },
    "dest": {
      "lineEndings": "\n"
    }
  },
  "processId": "iFhSZRvrz2vtFjcTSi9Qlg",
  "expirationDateTime": "2019-02-07T13:24:35.395Z",
  "state": "complete",
  "percentComplete": 100,
  "output": {
    "fileId": "eOsJIqI8aHkxVV0yJug"
  }
}
```

Error responses

Response when a plain text redactor process does not exist for the given processId:

```
HTTP 404
Content-Type: application/json
{
  errorCode: 'ResourceNotFound'
}
```

Response when provided markup work file cannot be parsed as JSON:

```
HTTP 200 OK
Content-Type: application/json
{
  "input": {
    "source": {
      "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
    },
    "markup": {
      "fileId": "vry3FPE0zQqYwhzndRccOQ"
    },
    "dest": {
      "lineEndings": "\n"
    }
  },
  "processId": "iFhSZRvrz2vtFjcTSi9Qlg",
  "state": "error",
  "errorCode": "InvalidJson",
  "percentComplete": 0,
  "expirationDateTime": "2019-02-07T13:24:35.395Z"
}
```

Response when markup JSON does not satisfy the Markup JSON Specification:

```
HTTP 200 OK
Content-Type: application/json
{
  "input": {
    "source": {
      "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
    },
    "markup": {
      "fileId": "vry3FPE0zQqYwhzndRccOQ"
    },
    "dest": {
      "lineEndings": "\n"
    }
  },
  "processId": "iFhSZRvrz2vtFjcTSi9Qlg",
  "state": "error",
  "errorCode": "InvalidMarkup",
  "percentComplete": 0,
  "expirationDateTime": "2019-02-07T13:24:35.395Z"
}
```

Response when markup refers to a page that does not exist in the source document:

```
{
  "input": {
    "source": {
      "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
    },
    "markup": {
      "fileId": "vry3FPE0zQqYwhzndRccOQ"
    },
    "dest": {
      "lineEndings": "\n"
    }
  },
  "processId": "iFhSZRvrz2vtFjcTSi9Qlg",
  "state": "error",
  "errorCode": "MarkRefersToNonExistentPage",
  "expirationDateTime": "2019-02-07T13:24:35.395Z"
}
```

Response when source document is a CAD file:

```
{
  "input": {
    "source": {
      "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
    },
    "markup": {
      "fileId": "vry3FPE0zQqYwhzndRccOQ"
    },
    "dest": {
      "lineEndings": "\n"
    }
  }
}
```

```
},
"processId": "iFhSZRvrz2vtFjcTSi9Qlg",
"state": "error",
"errorCode": "UnsupportedSourceDocument",
"expirationDateTime": "2019-02-07T13:24:35.395Z"
}
```

Redaction Creators

Introduction

The *redaction creators* REST API allows your application to automatically create redaction definitions for a document based on a given set of text-matching rules.

For application development in .NET, we recommend using the [PrizmDoc Server.NET SDK](#) instead of using the PrizmDoc Server REST API directly. See the [How to Create a Redacted PDF](#) topic in the .NET SDK documentation for an example of how to easily create redactions from a set of Regular Expression rules.

A redaction creator takes as input a source document and a set of text-matching regular expressions and produces as output a markup JSON file (or, if using the deprecated endpoints, a markup XML file). The output markup can then be used with the [markup burners REST API](#) to "burn" the redactions into a document.

Available URLs

URL	Description
POST /v2/redactionCreators	Creates and starts a new redaction creator process.
GET /v2/redactionCreators/{processId}	Gets the state of an existing redaction creator process.

Deprecated URLs

URL	Description
POST /PCCIS/V1/RedactionCreator	Creates and starts a new RedactionCreator.
GET /PCCIS/V1/RedactionCreator/{processId}	Gets the state and result of an existing RedactionCreator.

POST /v2/redactionCreators

Creates and starts a new redaction creator process.

A redaction creator represents a process that runs on the server which searches a document for text fragments matching specified rules and then, based on any matches, creates a new JSON Markup document containing redaction markup. The redaction markup that is created by this process can be used with the [markup burner](#) API to "burn" the redactions into a document.

The server process to create the redaction markup is started by this request, but a response is sent before the process is complete. Use the [GET /v2/redactionCreators/{processId}](#) URL below to get the state and results of an in-progress or completed redaction creator process.

There are two required inputs to create a RedactionCreator:

- One is a WorkFile object that represents the source document whose text will be searched, and
- One or more rules to match and redact the document text.

See the [work file](#) API topic for more information about a WorkFile.

A new markup document will be created that will contain the redaction markup. The new document will be made available by a new WorkFile ID.

Request

Request Headers

Name	Description
Content-Type	Must be <code>application/json</code>
Accusoft-Affinity-Token	The <code>affinityToken</code> of the work file specified by <code>input.documentFileId</code> . Required when server clustering is enabled.

Request Body

JSON object conforming to the following:

- `input`
 - `source` (Object) **Required.** Describes the document for which the markup will be created. This document will not be modified.
 - `fileId` (String) **Required.** The ID of the WorkFile that represents the source document whose text will be searched.
 - `rules` (Array) **Required.** The list of rules, each containing a Regular Expression to match the text and markup properties for any matches.
 - `find` (Object) **Required.** Describes the way to find matching text in the document.
 - `type` (String) **Required.** Must be `"regex"`. The type of the text match algorithm.
 - `pattern` (String) **Required, when type is "regex"**. The Regular Expression to match within the source document text using a [POSIX extended RE \(ERE\) or basic RE \(BRE\) syntax](#).
 - `redactWith` (Object) **Required.** Describes how to redact matching text.
 - `type` (String) **Required.** Must be `"RectangleRedaction"`. Type of redaction to be created for the matching text.
 - `reason` (String) Reason the content was redacted, displayed in the center of the redaction annotation.

NOTE: Only one of `"reason"` or `"reasons"` properties can be specified for a specific rule.
 - `reasons` (Array of strings) Reasons the content was redacted, displayed semicolon space separated in the center of the redaction annotation.

NOTE: Only one of the `"reason"` or `"reasons"` properties can be specified for a specific rule.
 - `fontColor` (String) Color of the text specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of `"#FF0000"` will create red text. Default is `"#FFFFFF"`.

- `fillColor` (String) Color of the redaction background specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of `"#FF0000"` will create a red background. Default is `"#000000"`.
- `borderColor` (String) Color of the border specified as a 6-character hexadecimal color string with a leading # sign. For example, a value of `"#FF0000"` will create a red border. Default is `"#000000"`.
- `borderThickness` (Integer) Thickness of the border in pixels. Should be greater than 0. Default is 1.

NOTE: Large `borderThickness` values may create borders which visually hide content from neighboring redacted text, however, this content will not be removed from the document during burning.

- `data` (Object) A property bag of user-defined values. Property values are only allowed to be strings.
- `minSecondsAvailable` (Integer) The minimum number of seconds this markup creator will remain available so you can GET its status. The actual lifetime may be longer. The default lifetime is defined by the `processIds.lifetime` [central configuration](#) parameter.

Successful Response

Response body

JSON object with the following properties:

- `input` (Object) A copy of the request body `input` object.
- `processId` (String) The id of the new redaction creator process.
- `expirationDateTime` (String) The date and time (in ISO 8601 Extended Format) when the `RedactionCreator` will be deleted.
- `state` (String) The current state of the redaction creation process running on the server. This will always be `"processing"` in this response.
- `percentComplete` (Integer) The percentage (0 – 100) complete of the redaction creation process. This will always be 0 in this response.
- `affinityToken` (String) Affinity token for this search context. Present when clustering is enabled.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
480	<code>"MissingInput"</code>	A required input value was not provided. See <code>errorDetails</code> in the response body.
480	<code>"InvalidInput"</code>	An invalid input value was used. See <code>errorDetails</code> in the response body.
480	<code>"InvalidInputCombination"</code>	An invalid combination of JSON properties was used. See <code>errorDetails</code> in the response body.
480	<code>"UnrecognizedInput"</code>	An unrecognized JSON property was provided. See <code>errorDetails</code> in the response body.
480	<code>"ResourceNotFound"</code>	Source document work file does not exist.
580	<code>"InternalError"</code>	The server encountered an internal error when handling the

Status Code	JSON <code>errorCode</code>	Description
		request.

Example

Request

```
POST /v2/redactionCreators
Content-Type: application/json
{
  "input": {
    "source": {
      "fileId": "NmFzioyWzFM5ADpthsuitw"
    },
    "rules": [{
      "find": {
        "type": "regex",
        "pattern": "wat"
      },
      "redactWith": {
        "type": "RectangleRedaction"
      }
    },
    {
      "find": {
        "type": "regex",
        "pattern": "[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"
      },
      "redactWith": {
        "type": "RectangleRedaction",
        "borderColor": "#FF0000",
        "fillColor": "#FF0000",
        "fontColor": "#000000",
        "reason": "Redacted",
        "data": {
          "author": "John Smith",
          "phone": "+1 123 456 789"
        }
      }
    }
  ]
}
```

Response

```
HTTP 200 OK
Content-Type: application/json
{
  "processId": "iFhSZRvrz2vtFjcTSi9Qlg",
  "expirationDateTime": "2019-02-07T13:24:35.395Z",
  "input": {
    "source": {
      "fileId": "NmFzioyWzFM5ADpthsuitw"
    },

```

```

"rules": [
  {
    "find": {
      "regex": "wat"
    },
    "redactWith": {
      "type": "RectangleRedaction"
    }
  },
  {
    "find": {
      "regex": "[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"
    },
    "redactWith": {
      "type": "RectangleRedaction",
      "borderColor": "#FF0000",
      "fillColor": "#FF0000",
      "fontColor": "#000000",
      "reason": "Redacted",
      "data": {
        "author": "John Smith",
        "phone": "+1 123 456 789"
      }
    }
  }
],
"state": "processing",
"percentComplete": 0
}

```

GET /v2/redactionCreators/{processId}

Gets the state and result of an existing redaction creator process created by [POST /v2/redactionCreators](#).

Requests can be sent to this URL repeatedly while `state` is `"processing"`.

When `state` is `"complete"`, the new markup document containing redactions will be made available by a new WorkFile ID in the `output.markupFileId`. See the [work file](#) API topic to find out how to download a WorkFile.

If the markup burning process encountered an error, the `state` property will be `"error"` and the `errorCode` property will contain an error code string.

Request

Request Headers

Name	Description
<code>Content-Type</code>	Must be <code>application/json</code>
<code>Accusoft-Affinity-Token</code>	The <code>affinityToken</code> of the work file specified by <code>input.documentFileId</code> . Required when server clustering is enabled.

URL Parameters

Parameter	Description
{processId}	The id of the redaction creator process.

Response body

JSON object with the following properties:

- `input` (Object) A copy of the [POST /v2/redactionCreators](#) request body `input` object we accepted to create the redaction creator process.
- `processId` (String) The id of the redaction creator process.
- `expirationDateTime` (String) The date and time (in ISO 8601 Extended Format) when the `RedactionCreator` will be deleted.
- `state` (String) The current state of the redaction creation process running on the server. The following values are allowed:
 - `"processing"` - The redaction creation is in progress.
 - `"complete"` - The redaction creation is completed.
 - `"error"` - The redaction creation returns an error.
- `percentComplete` (Integer) The percentage (0 – 100) complete of the redaction creation process. Will be 100 when `state` is `"complete"`.
- `errorCode` (String) An error code string if a problem occurred during the redaction creation process. Only present when `state` is `"error"`.
- `output` (Object) The object describes the created markup document. Only present when `state` is `"complete"`.
 - `markupFileId` (String) The ID of the new `WorkFile` that represents a new Markup document specifying the redactions.
- `affinityToken` (String) Affinity token for this search context. Present when clustering is enabled.

Error Responses and JSON Error Codes

Status Code	JSON <code>errorCode</code>	Description
404	<code>"ResourceNotFound"</code>	Redaction creator process specified by {processId} could not be found.
200	<code>"MarkupCreationError"</code>	The server encountered an error when creating the markup.
580	<code>"InternalError"</code>	The server encountered an internal error when handling the request.

Example

Request

```
GET /v2/redactionCreators/iFhSZRvrz2vtFjcTSi9Qlg
```

Successful Response

When the redaction creator process completes with no errors:

```
HTTP 200 OK
```

```
Content-Type: application/json
{
  "input": {
    "source": {
      "fileId": "NmFziowyWzFM5ADpthsuitw"
    },
    "rules": [
      {
        "find": {
          "regex": "wat"
        },
        "redactWith": {
          "type": "RectangleRedaction"
        }
      },
      {
        "find": {
          "regex": "[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"
        },
        "redactWith": {
          "type": "RectangleRedaction",
          "borderColor": "#FF0000",
          "fillColor": "#FF0000",
          "fontColor": "#000000",
          "reason": "Redacted",
          "data": {
            "author": "John Smith",
            "phone": "+1 123 456 789"
          }
        }
      }
    ]
  },
  "processId": "iFhSZRvrz2vtFjcTSi9Qlg",
  "expirationDateTime": "2019-02-07T13:24:35.395Z",
  "state": "complete",
  "percentComplete": 100,
  "output": {
    "markupFileId": "eOsJIqI8aHkxVV0yJug"
  }
}
```

Error responses

When a redaction converter process does not exist for the given processId:

```
HTTP 404
Content-Type: application/json
{
  errorCode: 'ResourceNotFound'
}
```

When source document is corrupted and Markup can not be created:

```
HTTP 200 OK
Content-Type: application/json
```

```
{
  "input": {
    "source": {
      "fileId": "NmFziyoyWzFM5ADpthsuitw"
    },
    "rules": [
      {
        "find": {
          "regex": "wat"
        },
        "redactWith": {
          "type": "RectangleRedaction"
        }
      },
      {
        "find": {
          "regex": "[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"
        },
        "redactWith": {
          "type": "RectangleRedaction",
          "borderColor": "#FF0000",
          "fillColor": "#FF0000",
          "fontColor": "#000000",
          "reason": "Redacted",
          "data": {
            "author": "John Smith",
            "phone": "+1 123 456 789"
          }
        }
      }
    ]
  },
  "processId": "iFhSZRvrz2vtFjcTSi9Qlg",
  "state": "error",
  "errorCode": "MarkupCreationError",
  "percentComplete": 0,
  "expirationDateTime": "2019-02-07T13:24:35.395Z"
}
```

POST /PCCIS/V1/RedactionCreator

NOTE: This URL has been deprecated and will be removed from the public API in a future release. Please use the newer [POST /v2/redactionCreators](#) instead.

Creates and starts a new RedactionCreator.

A RedactionCreator represents a process that runs on the server which searches a document for text matching a Regular Expression and then, based on any matches, creates a new Markup XML document containing redaction markup. The redaction Markup XML that is created by this process can be used with the [markup burner](#) API to "burn" the redactions into a document.

The server process to create the redaction markup is started by this request, but a response is sent before the process is complete. Use the [GET /PCCIS/V1/RedactionCreator/{processId}](#) URL below to get the state and results of an in-progress or completed RedactionCreator process.

There are two required inputs to create a RedactionCreator:

- One is a WorkFile object that represents the source document whose text will be searched, and
- One or more Regular Expressions to match the document text.

See the [work file](#) API topic for more information about a WorkFile.

A new Markup XML document will be created that will contain the redaction markup. The new document will be made available by a new WorkFile ID.

By default, RedactionCreator objects will be automatically deleted 20 minutes after they are created.

Request

Request Headers

Name	Description
Content-Type	Must be <code>application/json</code>
Accusoft-Affinity-Token	The <code>affinityToken</code> of the work file involved in the input to the process. Required when server clustering is enabled. Providing this value is important to ensuring the process will execute on the machine where the input work files actually exist. NOTE: If you do not provide the required <code>Accusoft-Affinity-Token</code> , the POST itself will succeed but the process itself will likely fail.

Request Body

In the request body, provide JSON containing the following properties:

- `input` (Object) Input to create the RedactionCreator.
 - `documentFileId` (String) **Required.** The ID of the WorkFile that represents the source document whose text will be searched. This document will not be modified.
 - `autoRedactionRegularExpressions` (Array of strings) **Required.** The Regular Expressions to match within the source document text. Multiple Regular Expressions provided in the array will be concatenated into a single Regular Expression using the format: "(regex1)|(regex2)...|(regexN)".
- `minSecondsAvailable` (Integer) The minimum number of seconds which this RedactionCreator must remain available. If not provided, a configurable default value is used. This value is ignored if it is shorter than the configurable value.

Response Body

If successful, this method returns JSON containing the following properties:

- `input` (Object) Input we accepted to create the RedactionCreator.
 - `documentFileId` (String) The ID of the WorkFile that represents the source document whose text will be searched. This document will not be modified. This is an echo of what was passed in by the request.
 - `autoRedactionRegularExpressions` (Array of Strings) The Regular Expressions to match within the source document text. This is an echo of what was passed in by the request.
- `expirationDateTime` (String) The date and time (in ISO 8601 Extended Format) when the RedactionCreator will be deleted.
- `processId` (String) The ID of the RedactionCreator.
- `state` (String) The current state of the markup burning process running on the server. This will always be **"processing"** in this response.
- `percentComplete` (Integer) The percentage (0 – 100) complete of the redaction creation process. This will always be **0** in this response.

- `errorCode` (String) An error code string if a problem occurred during the markup burning process. This will always be **null** in this response.
- `output` (Object)
 - `markupFileId` (String) The ID of the new WorkFile that represents the new redaction XML markup. This will always be **null** in this response.
- `affinityToken` (String) Affinity token for this search context. Present when clustering is enabled.

Error Responses

Status Code	Description
400	Bad Request, if <code>input.documentFileId</code> , <code>input.autoRedactionRegularExpressions</code> is missing or invalid format, or if <code>minSecondsAvailable</code> is not a number.
405	Method Not Allowed, if POST HTTP method is not used.

Examples

Request

```
POST prizmdoc_server_base_url/PCCIS/V1/RedactionCreator
Content-Type: application/json
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

```
{
  "input": {
    "documentFileId": "ek5Zb123oYHSUEVx1bUrVQ",
    "autoRedactionRegularExpressions": [
      "[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"
    ]
  },
  "minSecondsAvailable": 60
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "input": {
    "documentFileId": "ek5Zb123oYHSUEVx1bUrVQ",
    "autoRedactionRegularExpressions": [
      "[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"
    ]
  },
  "expirationDateTime": "2014-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "processing",
  "percentComplete": 0,
  "errorCode": null,
  "output": null
}
```

GET /PCCIS/V1/RedactionCreator/{processId}

NOTE: This URL has been deprecated and will be removed from the public API in a future release. Please use the newer [POST /v2/redactionCreators](#) and [GET /v2/redactionCreators/{processId}](#) instead.

Gets the state and result of an existing RedactionCreator.

Requests can be sent to this URL repeatedly while `state` is "processing".

When `state` is "complete", the new markup XML document containing redactions will be made available by a new WorkFile ID in the `output.markupFileId`. See the [work file](#) API topic to find out how to download a WorkFile.

If the markup burning process encountered an error, the `state` property will be "error", the `errorCode` property will contain an error code string and `output` will be null.

Request

Request Headers

Name	Description
Content-Type	Must be <code>application/json</code>
Accusoft-Affinity-Token	The <code>affinityToken</code> of the work file specified by <code>input.documentFileId</code> . Required when server clustering is enabled.

URL Parameters

Parameter	Description
{processId}	The id of the process.

Response Body

If successful, this method returns JSON containing the following properties:

- `input` (Object) Input we accepted to create the MarkupBurner.
 - `documentFileId` (String) The ID of the WorkFile that represents the source document whose text will be searched. This document will not be modified.
 - `autoRedactionRegularExpressions` (Array of Strings) The Regular Expressions to match within the source document text.
- `expirationDateTime` (String) The date and time (in ISO 8601 Extended Format) when the RedactionCreator will be deleted.
- `processId` (String) The ID of the RedactionCreator.
- `state` (String) The current state of the redaction creation process running on the server. The following values are allowed:
 - "processing" - The redaction creation is in progress.
 - "complete" - The redaction creation is completed.
 - "error" - The redaction creation returns an error.
- `percentComplete` (Integer) The percentage (0 – 100) complete of the redaction creation process.
- `errorCode` (String) An error code string if a problem occurred during the redaction creation process.

- `output` (Object)
 - `markupFileId` (String) The ID of the new WorkFile that represents a new XML Markup document specifying the redactions.
- `affinityToken` (String) Affinity token for this search context. Present when clustering is enabled.

Error Responses

Status Code	Description
404	Not Found, if <code>{processId}</code> does not exist.
405	Method Not Allowed, if GET HTTP method is not used.

Examples

Request

```
GET prizmdoc_server_base_url/PCCIS/V1/RedactionCreator/ElkNzWtrUJp4rXI5YnLUgw
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

Response when the state is "processing"

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentFileId": "ek5Zb123oYHSUEVx1bUrVQ",
    "autoRedactionRegularExpressions": [
      "[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"
    ]
  },
  "expirationDateTime": "2014-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "processing",
  "percentComplete": 0,
  "errorCode": null,
  "output": null
}
```

Response when the state is "complete"

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentFileId": "ek5Zb123oYHSUEVx1bUrVQ",
    "autoRedactionRegularExpressions": [
      "[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"
    ]
  }
}
```

```

    },
    "expirationDateTime": "2014-12-17T20:38:39.796Z",
    "processId": "ElkNzWtrUJp4rXI5YnLUgw",
    "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
    "state": "complete",
    "percentComplete": 100,
    "errorCode": null,
    "output": {
      "markupFileId": "vry3FPPE0zQqYwhzndRccOQ"
    }
  }
}

```

Response when the state is "error" because the work file that represents the source document whose text will be searched could not be found

```

HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentFileId": "ek5Zb123oYHSUEVx1bUrVQ",
    "markupFileId": "aQ1BdViqmUisBuevJKO9Sw"
  },
  "expirationDateTime": "2014-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "error",
  "percentComplete": 0,
  "errorCode": "DocumentFileIdDoesNotExist"
}

```

Search Contexts

Introduction

The *search context* and *search task* REST APIs allow your application to perform server-side searching and text retrieval of a document.

A search context contains a collection of *records* of full-page text data, one record per page.

Available URLs

URL	Description
POST /v2/searchContexts	Creates a new search context.
GET /v2/searchContexts/{contextId}	Gets information about a search context.
DELETE /v2/searchContexts/{contextId}	Deletes a search context.
PUT /v2/searchContexts/{contextId}/records	Uploads previously extracted text records to a context, when the context uses <code>input.source</code> of "upload".

URL	Description
POST /v2/searchContexts/{contextId}/completed	Marks all previously extracted text records as uploaded, when the context uses <code>input.source</code> of "upload".
GET /v2/searchContexts/{contextId}/records	Gets full-page text data (records) for a specified set of pages.

POST /v2/searchContexts

Creates a search context which will eventually hold a set of full-page text records for a source document.

After a successful POST to create the search context, we immediately begin a background process to extract the text records using a [work file](#) you specified in the POST (via `input.fileId`). As we extract pages of text, new records will become available for you to [GET](#). The search context `state` will change from "processing" to "complete" when there are no more records to extract.

Request

Request Headers

Name	Description
<code>Content-Type</code>	Must be <code>application/json</code>
<code>Accusoft-Affinity-Token</code>	The <code>affinityToken</code> of the work file specified by <code>input.fileId</code> . Required when server clustering is enabled and <code>input.source</code> is "workFile".

Request Body

- `input`
 - `documentIdentifier` (String) **Required.** Your own unique identifier for the source document. *It is crucial that you use a unique value for each unique document, otherwise, the returned text for a document will not be correct.*
 - `source` (String) **Required.** The following values are allowed:
 - "workFile" - Indicates that the server should use an existing [work file](#) and extract the text from it.
 - "upload" - Indicates that the user would like to upload previously extracted text to the server to be used for search.
 - `fileId` (String) **Required with `source: "workfile"`.** The id of the [work file](#) to extract text records from.
 - `password` (String) Password to open the source document, when using a `source` of "workFile".
- `minSecondsAvailable` (Integer) The minimum number of seconds this search context will remain available. The actual lifetime may be longer. The default lifetime is defined by the `processIds.lifetime` [central configuration](#) parameter.

Successful Response

Response Body

JSON with metadata about the created search context. You can check for changes to this metadata with additional [GET requests](#).

- `input` (Object) Input we accepted to create the search context.
- `contextId` (String) Unique id for this search context.
- `affinityToken` (String) Affinity token for this search context. Present when clustering is enabled.
- `state` (String) State of acquiring text records for the given `input.documentIdentifier`.
 - "processing" - The server is acquiring text records. No further actions are needed.
 - "awaitingInput" - The server is waiting for text to be uploaded by client. The client should take action and provide all required text records. This state only occurs when the client created a `searchContext` using `input.source: "upload"`. Note that based on the provided `documentIdentifier`, the server may skip this state and go directly to "processing" or "complete" if it determines that it already has the required data.
 - "complete" - All text records have been acquired.
 - "error" - There was a problem acquiring text records.
- `percentComplete` (Integer) Percentage of text records which have been acquired (from 0 to 100).
- `expirationDateTime` (String) Currently planned date and time when the search context resource will expire and no longer be available for use. This time may be extended if we have need to keep using the data (for example, if there are [search tasks](#) executing against this context). Format is [RFC 3339 Internet Date/Time profile of ISO 8601](#), e.g. "2016-11-05T08:15:30.494Z".
- `errorCode` (String) Descriptive error code. Present when `state` is "error".
- `errorDetails` (Object) Additional error details, if any. May be present when `errorCode` is present.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
400	"MissingInput"	Can occur when clustering is enabled and an <code>Accusoft-Affinity-Token</code> request header was not provided.
480	"MissingInput"	A required input value was not provided. See <code>errorDetails</code> in the response body.
480	"InvalidInput"	An invalid input value was used. See <code>errorDetails</code> in the response body.
580	"InternalError"	The server encountered an internal error when handling the request.

Examples

Creating a `searchContext` using a workfile:

Request

```
POST prizmdoc\_server\_base\_url/v2/searchContexts
Content-Type: application/json
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

{
  "input": {
    "documentIdentifier": "your-own-unique-identifier-for-the-source-document",
    "source": "workFile",
    "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
  }
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentIdentifier": "your-own-unique-identifier-for-the-source-document",
    "source": "workFile",
    "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
  },
  "contextId": "ElkNzWtrUJp4rXI5YnLUgw",
  "state": "processing",
  "percentComplete": 0,
  "expirationDateTime": "2016-12-17T20:38:39.796Z",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

Creating a `searchContext` using previously extracted text:

Note that it is recommended that you use the `Accusoft-Affinity-Hint` header here when working in multi-server mode, so that multiple contexts created for the same document can be routed to the same server when possible.

Request

```
POST prizmdoc\_server\_base\_url/v2/searchContexts
Content-Type: application/json
Accusoft-Affinity-Hint: "your-own-unique-identifier-for-the-source-document"

{
  "input": {
    "documentIdentifier": "your-own-unique-identifier-for-the-source-document",
    "source": "upload"
  },
  "minSecondsAvailable": 1200
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentIdentifier": "your-own-unique-identifier-for-the-source-document",
    "source": "upload"
  },
  "contextId": "ElkNzWtrUJp4rXI5YnLUgw",
  "state": "awaitingInput",
  "percentComplete": 0,
  "expirationDateTime": "2016-12-17T20:38:39.796Z",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

GET /v2/searchContexts/{contextId}

Gets information about a search context.

Request

URL Parameters

Parameter	Description
{contextId}	The <code>contextId</code> which identifies the resource.

Request Headers

Name	Description
Accusoft-Affinity-Token	The <code>affinityToken</code> of the search context. Required when server clustering is enabled.

Successful Response

Response Body

JSON with current metadata about the search context.

- `input` (Object) Input we accepted to create the search context.
- `contextId` (String) Unique id for this search context.
- `affinityToken` (String) Affinity token for this search context. Present when clustering is enabled.
- `state` (String) State of acquiring text records for the given `input.documentIdentifier`.
 - "processing" - The server is acquiring text records. No further actions are needed.
 - "awaitingInput" - The server is waiting for text to be uploaded by client. The client should take action and provide all required text records. This state only occurs when the client created a `searchContext` using `input.source: "upload"`. Note that based on the provided `documentIdentifier`, the server may skip this state and go directly to "processing" or "complete" if it determines that it already has the required data.
 - "complete" - All text records have been acquired.
 - "error" - There was a problem acquiring text records.
- `percentComplete` (Integer) Percentage of text records which have been acquired (from 0 to 100).
- `expirationDateTime` (String) Currently planned date and time when the search context resource will expire and no longer be available for use. This time may be extended if we have need to keep using the data (for example, if there are [search tasks](#) executing against this context). Format is [RFC 3339 Internet Date/Time profile of ISO 8601](#), e.g. "2016-11-05T08:15:30-05:00".
- `errorCode` (String) Descriptive error code. Present when `state` is "error".
- `errorDetails` (Object) Additional error details, if any. May be present when `errorCode` is present.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
400	"MissingInput"	Can occur when clustering is enabled and an <code>Accusoft-Affinity-Token</code> request header was not provided.
580	"InternalError"	The server encountered an internal error when handling the request.

Examples

Request

```
GET prizmdoc_server_base_url/v2/searchContexts/ElkNzWtrUJp4rXI5YnLUgw
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

Response when the state is still "processing"

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentIdentifier": "your-own-unique-identifier-for-the-source-document",
    "source": "workFile",
    "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
  },
  "contextId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "processing",
  "percentComplete": 47,
  "expirationDateTime": "2016-12-17T20:38:39.796Z"
}
```

Response when the state is "complete"

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentIdentifier": "your-own-unique-identifier-for-the-source-document",
    "source": "workFile",
    "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
  },
  "contextId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "complete",
  "percentComplete": 100,
  "expirationDateTime": "2016-12-17T20:38:39.796Z"
}
```

Response when the state is "error" because the work file could not be found

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentIdentifier": "your-own-unique-identifier-for-the-source-document",
    "source": "workFile",
    "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
  },
  "contextId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "error",
  "errorCode": "ResourceNotFound",
  "errorDetails": {
    "in": "searchContext",
    "at": "input.fileId"
  },
  "expirationDateTime": "2016-12-17T20:38:39.796Z"
}
```

Response when the source document required a password but no password was provided

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentIdentifier": "your-own-unique-identifier-for-the-source-document",
    "source": "workFile",
    "fileId": "ek5Zb123oYHSUEVx1bUrVQ"
  },
  "contextId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "error",
  "errorCode": "InvalidPassword",
  "errorDetails": {
    "in": "searchContext",
    "at": "input.password"
  },
  "expirationDateTime": "2016-12-17T20:38:39.796Z"
}
```

Response when the source document required a password but the wrong password was provided

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "documentIdentifier": "your-own-unique-identifier-for-the-source-document",
    "source": "workFile",
    "fileId": "ek5Zb123oYHSUEVx1bUrVQ",

```

```
    "password": "wrong-password"
  },
  "contextId": "ElkNzWtrUJp4rXI5YnLUgw",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "error",
  "errorCode": "InvalidPassword",
  "errorDetails": {
    "in": "searchContext",
    "at": "input.password"
  },
  "expirationDateTime": "2016-12-17T20:38:39.796Z"
}
```

DELETE /v2/searchContexts/{contextId}

Deletes a search context. Further requests using this `contextId` will return errors.

Request

URL Parameters

Parameter	Description
{contextId}	The <code>contextId</code> which identifies the resource.

Request Headers

Name	Description
Accusoft-Affinity-Token	The <code>affinityToken</code> of the search context. Required when server clustering is enabled.

Successful Response

This request returns no body in the response when successful.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	"Not Found"	No search context with the provided <code>contextId</code> could be found.
400	"MissingInput"	Can occur when clustering is enabled and an <code>Accusoft-Affinity-Token</code> request header was not provided.
580	"InternalError"	The server encountered an internal error when handling the request.

Examples

Request

```
DELETE prizmdoc_server_base_url/v2/searchContexts/ElkNzWtrUJp4rXI5YnLUgw
```

```
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

Response

```
HTTP/1.1 204 No Content
```

PUT /v2/searchContexts/{contextId}/records

This URL is used to upload one or more previously extracted text records to a search context.

Note that this is only necessary when creating a `searchContext` using `input.source` of "upload" and receive a state of "awaitingInput".

Request

URL Parameters

Parameter	Description
{contextId}	The <code>contextId</code> which identifies the resource.

Request Headers

Name	Description
Content-Type	Must be <code>application/json</code>
Accusoft-Affinity-Token	The <code>affinityToken</code> of the search context. Required when server clustering is enabled.

Request Body

NOTE: Since this is previously extracted text being uploaded, the body of the request corresponds to the body of the response on `GET /v2/searchContexts/{contextId}/records`.

- `pages []` (Array of Objects) Array of full-page text record objects for the requested pages. Note that the order of the records is not guaranteed; you must use the `number` property of each returned item to know its page index. Items may contain:
 - `number` (Integer) **Required.** Page index (zero-indexed page number). The property is named simply `number` for backwards compatibility reasons.
 - `text` (String) Page text. Either `text` or `errorCode` is required.
 - `errorCode` (String) A value indicating there was a problem with the page text. Either `text` or `errorCode` is required.
 - `width` (Number) **Required with `text`.** Page width.
 - `height` (Number) **Required with `text`.** Page height.
 - `rectangles []` (Array of Arrays) **Required with `text`.** Bounding boxes for individual glyphs on the page. Each item will contain four numbers:
 - `[0]` (Number) Distance from the left edge of the page to the left edge of the glyph bounding box.

- [1] (Number) Distance from the top edge of the page to the top edge of the glyph bounding box.
- [2] (Number) Width of the glyph bounding box.
- [3] (Number) Height of the glyph bounding box.
- markup[] (Array of Objects) Objects describing hyperlinks, if any. Each item may contain:
 - changeType (String) Value will always be "Add".
 - markType (String) Value will always be "DocumentHyperlink".
 - properties (Object) Properties of the hyperlink.
 - href (String) Destination URL.
 - rectangle (Object) Dimensions of the hyperlink bounding box on the page.
 - x (Number) Distance from the left edge of the page to the left edge of the hyperlink bounding box.
 - y (Number) Distance from the top edge of the page to the top edge of the hyperlink bounding box.
 - width (Number) Width of the hyperlink bounding box.
 - height (Number) Height of the hyperlink bounding box.
 - borderThickness (Number) Border thickness which should be applied.
 - borderHorizontalRadius (Number) Horizontal border radius which should be applied.
 - borderVerticalRadius (Number) Vertical border radius which should be applied.
 - borderOpacity (Integer) Border opacity which should be applied. Value will be from 0 to 255, where 0 represents fully transparent and 255 represents fully opaque.

Successful Response

This request returns no body in the response when successful.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
400	"MissingInput"	Can occur when clustering is enabled and an <code>Accusoft-Affinity-Token</code> request header was not provided.
480	"MissingInput"	A required input value was not provided. See <code>errorDetails</code> in the response body.
480	"InvalidInput"	An invalid input value was used. See <code>errorDetails</code> in the response body.
480	"ResourceNotUsable"	The search context is in a <code>state</code> of "error", or has otherwise become unusable.
480	"IncorrectUsage"	The <code>state</code> of the search context is not correct. See <code>errorDetails</code> in the response body.
580	"InternalError"	The server encountered an internal error when handling the request.

Examples

Request

```
PUT prizmdoc_server_base_url/v2/searchContexts/ElkNzWtrUJp4rXI5YnLUgw/records
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

```
Content-Type: application/json

{
  "pages": [{
    "number": 1,
    "text": "the text to be searched",
    "width": 147,
    "height": 349
    "rectangles": [
      [ 23.6, 767.75, 15.01, 23.08 ],
      ...
    ]
  }, ...]
}
```

Responses

When the data was successfully accepted

```
HTTP/1.1 200 OK
```

When the search context is not awaiting input

```
HTTP/1.1 480 IncorrectUsage
Content-Type: application/json

{
  "errorCode": "IncorrectUsage",
  "errorDetails": {
    "in": "searchContext",
    "at": "state",
    "actual": "processing",
    "expected": {
      "value": "awaitingInput"
    }
  }
}
```

POST /v2/searchContexts/{contextId}/completed

This URL is used to let the server know that all previously extracted records have been uploaded.

Note that this is only necessary when creating a `searchContext` using `input.source` of `"upload"` and receive a state of `"awaitingInput"`.

The provided records should make up a set of contiguous page records (e.g. `[1, 2, 3, 4, 5]` and not `[1, 2, 3, 5, 27]`), and if any pages are missing from the set, the context will not be allowed to complete successfully.

Request

URL Parameters

Parameter	Description
{contextId}	The contextId which identifies the resource.

Request Headers

Name	Description
Accusoft-Affinity-Token	The affinityToken of the search context. Required when server clustering is enabled.

Request Body

This request has no body.

Successful Response

This request returns no body in the response when successful.

Error Responses

Status Code	JSON errorCode	Description
400	"MissingInput"	Can occur when clustering is enabled and an Accusoft-Affinity-Token request header was not provided.
480	"ResourceNotUsable"	The search context is in a state of "error", or has otherwise become unusable.
480	"IncorrectUsage"	The state of the search context is not correct. See errorDetails in the response body.
480	"MissingRecords"	A non-contiguous set of pages was present at the time that this request was made.
580	"InternalError"	The server encountered an internal error when handling the request.

Examples

Request

```
POST prizmdoc_server_base_url/v2/searchContexts/ElkNzWtrUJp4rXI5YnLUgw/completed
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

Responses

When the context is successfully completed

```
HTTP/1.1 200 OK
```

When a non-contiguous range of pages is provided (e.g. [1, 2, 3, 5, 27])

```
HTTP/1.1 480 MissingRecords
Content-Type: application/json

{
  "errorCode": "MissingRecords"
}
```

When the state of the context is "error"

```
HTTP/1.1 480 ResourceNotUsable
Content-Type: application/json

{
  "errorCode": "ResourceNotUsable"
}
```

When the context is not awaiting input

```
HTTP/1.1 480 IncorrectUsage
Content-Type: application/json

{
  "errorCode": "IncorrectUsage",
  "errorDetails": {
    "in": "searchContext",
    "at": "state",
    "actual": "processing",
    "expected": {
      "enum": ["awaitingInput", "complete"]
    }
  }
}
```

GET /v2/searchContexts/{contextId}/records?pages={pages}

Gets full-page text data (records) for a specified set of pages.

Request

URL Parameters

Parameter	Description
{contextId}	The contextId which identifies the resource.
{pages}	Required. A set of comma-delimited page indices (zero-indexed page numbers) and/or

Parameter	Description
	hyphenated page index ranges for which you want the full-page text data (<i>records</i>). See more below .

pages

The pages parameter accepts one or more zero-indexed page numbers (page indices). Between commas, you can specify individual pages (like 0), closed page ranges (like 0-3), and open-ended page ranges (like 3-, which means page index 3 through the end of the document).

Here are some examples:

Example	Description
pages=0	Get the text data for page index 0.
pages=5	Get the text data for page index 5.
pages=0-5	Get the text data for page indices 0-5.
pages=3-	Get the text data for page indices 3 through the end of the document.
pages=0-	Get the text data for all pages (page index 0 through the end of the document).
pages=1-	Get the text data for all but the first page (page index 1 through the end of the document).
pages=0, 2, 5, 9	Get the text data for page indices 0, 2, 5, and 9.
pages=2, 4-5, 7-	Get the text data for page indices 2, 4 through 5, and 7 through the end of the document.

Request Headers

Name	Description
Accusoft-Affinity-Token	The <code>affinityToken</code> of the search context. Required when server clustering is enabled.

Successful Response

JSON containing full-page text records for the requested pages.

- pages []** (Array of Objects) **Always present.** Array of full-page text record objects for the requested pages. Note that the order of the records is not guaranteed; you must use the `number` property of each returned item to know its page index. Items may contain:
 - `number` (Integer) **Always present.** Page index (zero-indexed page number). The property is named simply `number` for backwards compatibility reasons.
 - `text` (String) Page text.
 - `errorCode` (String) A descriptive page-level error code (such as "CouldNotGetPageData") if there was a problem getting data for the page.
 - `width` (Number) Page width.
 - `height` (Number) Page height.
 - `rectangles []` (Array of Arrays) Bounding boxes for individual glyphs on the page. Each item will contain four numbers:
 - `[0]` (Number) Distance from the left edge of the page to the left edge of the glyph bounding box.

- [1] (Number) Distance from the top edge of the page to the top edge of the glyph bounding box.
 - [2] (Number) Width of the glyph bounding box.
 - [3] (Number) Height of the glyph bounding box.
 - markup[] (Array of Objects) Objects describing hyperlinks, if any. Each item may contain:
 - changeType (String) Value will always be "Add".
 - markType (String) Value will always be "DocumentHyperlink".
 - properties (Object) Properties of the hyperlink.
 - href (String) Destination URL.
 - rectangle (Object) Dimensions of the hyperlink bounding box on the page.
 - x (Number) Distance from the left edge of the page to the left edge of the hyperlink bounding box.
 - y (Number) Distance from the top edge of the page to the top edge of the hyperlink bounding box.
 - width (Number) Width of the hyperlink bounding box.
 - height (Number) Height of the hyperlink bounding box.
 - borderThickness (Number) Border thickness which should be applied.
 - borderHorizontalRadius (Number) Horizontal border radius which should be applied.
 - borderVerticalRadius (Number) Vertical border radius which should be applied.
 - borderOpacity (Integer) Border opacity which should be applied. Value will be from 0 to 255, where 0 represents fully transparent and 255 represents fully opaque.
- errorCode (String) Descriptive error code. Present if there was a general problem getting all of the requested data.
- errorDetails (Object) Present if there are additional error details.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404		No search context exists for the <code>{contextId}</code> given in the URL. It may have expired, or it may have never existed.
400	"MissingInput"	Can occur when clustering is enabled and an <code>Accusoft-Affinity-Token</code> request header was not provided.
480	"MissingInput"	A required input was missing. See the <code>errorDetails</code> for more information.
480	"InvalidSyntax"	Can occur when the <code>pages</code> query string parameter is set to a value we cannot understand.
480	"ResourceNotUsable"	Can occur when the search context is in a <code>state</code> of "error". You may be able to get more information from a <code>GET /v2/searchContexts/{contextId}</code> .
580	"InternalError"	The server encountered an internal error when handling the request.

Examples

When all data is returned successfully

Request records for pages 0 through 9:

```


```

```
GET prizmdoc\_server\_base\_url/v2/searchContexts/ElkNzWtrUJp4rXI5YnLUgw/records?pages=0-9
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

Successful response (where ... indicates that data has been omitted for brevity):

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "pages": [
    {
      "number": 0,
      "text": "the page text",
      "width": 648.00,
      "height": 828.00,
      "rectangles": [
        [
          202.25,
          135.05,
          27.00,
          73.26
        ],
        [
          229.25,
          135.05,
          30.00,
          73.26
        ],
        ...
      ]
    },
    {
      "markup": [
        {
          "changeType": "Add",
          "markType": "DocumentHyperlink",
          "properties": {
            "rectangle": {
              "height": 14.71,
              "width": 86.20,
              "y": 73.50,
              "x": 71.31
            },
            "borderHorizontalRadius": 0.0,
            "borderVerticalRadius": 0.0,
            "borderThickness": 0.0,
            "href": "http://www.google.com/",
            "borderOpacity": 255
          }
        },
        ...
      ]
    },
    ...
  ]
}
```

When the data stream is interrupted

Because this URL may return large amounts of data, we progressively stream data to the HTTP response. As such, it is possible that we encounter a data streaming error after we have sent HTTP 200. When this happens, we will close the JSON with a top-level `errorCode` of `"DataStreamInterruption"`, like so:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "pages": [...],
  "errorCode": "DataStreamInterruption"
}
```

When out-of-range, non-existent pages are requested

If you request a set of pages that include non-existent pages beyond the length of the document, we will include whatever actual `pages` we can, but we will also add a top-level `errorCode` of `"RequestedPagesOutOfRange"` with the actual `documentPageCount` within an `errorDetails` object, like so:

```
GET prizmdoc\_server\_base\_url/v2/searchContexts/ElkNzWtrUJp4rXI5YnLUgw/records?pages=0-9
Content-Type: application/json
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "pages": [...],
  "errorCode": "RequestedPagesOutOfRange",
  "errorDetails": {
    "documentPageCount": 3
  }
}
```

When data cannot be extracted from some pages

The `pages` array will contain one item for each requested page that actually exists. If we are unable to obtain data for a particular page, we will include an item in the `pages` array that contains the page `number` and a page-specific `errorCode` of `"CouldNotGetPageData"`, like so:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "pages": [
    {
      "number": 0,
      "text": "Once upon a time...",
      "width": 612.00,
      "height": 792.00,
      "rectangles": [...]
```



```

    },
    {
      "number": 1,
      "errorCode": "CouldNotGetPageData"
    },
    {
      "number": 2,
      "errorCode": "CouldNotGetPageData"
    },
    {
      "number": 3,
      "text": "and then, she said to the dragon...",
      "width": 612.00,
      "height": 792.00,
      "rectangles": [...]
    }
  ]
}

```

Search Tasks

Introduction

The [search context](#) and [search task](#) REST APIs allow your application to perform server-side searching and text retrieval of a document.

A search task represents an asynchronous full-text search of a document (via a [search context](#)) and yields results as they become available.

Available URLs

URL	Description
POST /v2/searchTasks	Starts an asynchronous full-text search against a search context .
POST /v2/viewingSessions/{viewingSessionId}/searchTasks	Starts an asynchronous full-text search against a viewing session's source document.
GET /v2/searchTasks/{processId}	Gets information about a search task.
GET /v2/searchTasks/{processId}/results	Gets available search results.
DELETE /v2/searchTasks/{processId}	Cancels a search task.

POST /v2/searchTasks

Starts an asynchronous full-text search against a [search context](#).

After a successful POST to create the search task, we immediately begin a background process to start populating search results for you to [GET](#). You do not need to wait for the full set of results to be available; you can start [retrieving partial search results](#) as soon as they are available. Once the full text of the document has been searched and no more results will be added, the search task state will change from "processing" to "complete".

Request

Request Headers

Name	Description
Content-Type	Must be <code>application/json</code>
Accusoft-Affinity-Token	The <code>affinityToken</code> of the search context specified by <code>input.contextId</code> . Required when server clustering is enabled.

Request Body

- `input`
 - `contextId` (String) **Required**. Identifies the [search context](#) which holds the full-text data to search.
 - `searchTerms` [] (Array of Objects) **Required and must contain at least one item**. Each item must be an object which conforms to one of the following:
 - *Simple (finds all occurrences of a single regex pattern):*
 - `type`: "simple" (String) **Required**. Must be set to "simple" to indicate this is a simple term object.
 - `pattern` (String) **Required**. Regular expression to search for, using a [JavaScript-flavored regular expression string](#).
 - `caseSensitive` (Boolean) Determines whether we consider case when matching this term. Default is `false`.
 - `contextPadding` (Integer) Maximum number of characters to include both before and after the search result in the returned `context` string. For example, a value of 25 would allow up to 25 preceding and 25 following characters of content. Default is 25.
 - `termId` (String) Optional id of your choosing which, if provided, will be included as a `termId` property on each search result produced by this term. When used, we do not enforce uniqueness; it is your responsibility to use a unique `termId` for each term.
 - *Proximity (finds all occurrences of multiple regex patterns which are near each other):*
 - `type`: "proximity" (String) **Required**. Must be set to "proximity" to indicate this is a proximity term object.
 - `subTerms` [] (Array of Objects) **Required and must contain at least two items**. Each item may contain:
 - `pattern` (String) **Required**. Regular expression for this particular term, using a [JavaScript-flavored regular expression string](#).
 - `caseSensitive` (Boolean) Determines whether we consider case when matching this term. Default is `false`.

- `distance` (Integer) **Required.** Maximum number of words allowed between any two consecutive search terms.
 - `contextPadding` (Integer) Maximum number of characters to include both before and after the search result in the returned `context` string. For example, a value of 25 would allow up to 25 preceding and 25 following characters of content. Default is 25.
 - `termId` (String) Optional id of your choosing which, if provided, will be included as a `termId` property on each search result produced by this term. When used, we do not enforce uniqueness; it is your responsibility to use a unique `termId` for each term.
- `minSecondsAvailable` (Integer) The minimum number of seconds this search task will remain available. The actual lifetime may be longer. The default lifetime is defined by the `processIds.lifetime` [central configuration](#) parameter.

Successful Response

Response Body

JSON with metadata about the created search task.

- `input` (Object) Input we accepted to create the search task.
- `processId` (String) Unique id for this search task.
- `affinityToken` (String) Affinity token for this search task. Present when clustering is enabled.
- `state` (String) State of getting search results.
 - "processing" - The search is still being executed. Additional results may become available.
 - "complete" - The search is complete. No additional results will become available.
 - "error" - There was a problem performing the search. No additional results will become available.
- `percentComplete` (Integer) Percentage of the document text which has been searched (from 0 to 100).
- `expirationDateTime` (String) Currently planned date and time when the search task resource will expire and no longer be available for use. Format is [RFC 3339 Internet Date/Time profile of ISO 8601](#), e.g. "2016-11-05T08:15:30.494Z".

Error Responses

Status Code	JSON <code>errorCode</code>	Description
400	"MissingInput"	Can occur when clustering is enabled and an <code>Accusoft-Affinity-Token</code> request header was not provided.
480	"MissingInput"	A required input value was not provided. See <code>errorDetails</code> in the response body.
480	"InvalidInput"	An invalid input value was used. See <code>errorDetails</code> in the response body.
480	"MissingInputForSimpleTerm"	An invalid input value was used in a "simple" term object. See <code>errorDetails</code> in the response body.
480	"InvalidInputForSimpleTerm"	An invalid input value was used in a "simple" term object. See <code>errorDetails</code> in the response body.
480	"MissingInputForProximityTerm"	An invalid input value was used in a "proximity" term object. See <code>errorDetails</code> in the response body.
480	"InvalidInputForProximityTerm"	An invalid input value was used in a "proximity" term object. See <code>errorDetails</code> in the response body.
480	"ResourceNotFound"	Can occur when the search context specified by <code>contextId</code> could not be found. See <code>errorDetails</code> in the response body.
480	"ResourceNotUsable"	Can occur when the search context specified by <code>contextId</code> is not usable. See <code>errorDetails</code> in the response body.
580	"InternalError"	The server encountered an internal error when handling the request.

Example

Request

```
POST prizmdoc\_server\_base\_url/v2/searchTasks
Content-Type: application/json
Accusoft-Affinity-Token: eJN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

{
  "input": {
    "contextId": "ElkNzWtrUJp4rXI5YnLUgw",
    "searchTerms": [{
      "type": "simple",
      "pattern": "quick"
    }]
  }
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "contextId": "ElkNzWtrUJp4rXI5YnLUgw",
    "searchTerms": [{
      "type": "simple",
      "pattern": "quick",
      "caseSensitive": false,
      "contextPadding": 25
    }]
  },
  "processId": "pR5X6nPDgMwat6cxlmn0Q3",
  "state": "processing",
  "percentComplete": 0,
  "expirationDateTime": "2016-12-17T20:38:39.796Z"
}
```

Additional Examples

For more examples of how to construct different searches, see [Example Searches](#).

POST /v2/viewingSessions/{viewingSessionId}/searchTasks

Starts an asynchronous full-text search against a viewing session's source document.

After a successful POST to create the search task, we immediately begin a background process to start populating search results for you to [GET](#). You do not need to wait for the full set of results to be available; you can start [retrieving partial search results](#) as soon as they are available. Once the full text of the document has been searched and no more results will be added, the search task state will change from "processing" to "complete".

Request

Request Headers

Name	Description
Content-Type	Must be <code>application/json</code>

Request Body

- `input`
 - `searchTerms[]` (Array of Objects) **Required and must contain at least one item.** Each item must be an object which conforms to one of the following:
 - **Simple** (finds all occurrences of a single regex pattern):
 - `type: "simple"` (String) **Required.** Must be set to "simple" to indicate this is a simple term object.
 - `pattern` (String) **Required.** Regular expression to search for, using a [JavaScript-flavored regular expression string](#).
 - `caseSensitive` (Boolean) Determines whether we consider case when matching this term. Default is `false`.
 - `contextPadding` (Integer) Maximum number of characters to include both before and after the search result in the returned `context` string. For example, a value of 25 would allow up to 25 preceding and 25 following characters of content. Default is 25.
 - `termId` (String) Optional id of your choosing which, if provided, will be included as a `termId` property on each search result produced by this term. When used, we do not enforce uniqueness; it is your responsibility to use a unique `termId` for each term.
 - **Proximity** (finds all occurrences of multiple regex patterns which are near each other):
 - `type: "proximity"` (String) **Required.** Must be set to "proximity" to indicate this is a proximity term object.
 - `subTerms[]` (Array of Objects) **Required and must contain at least two items.** Each item may contain:
 - `pattern` (String) **Required.** Regular expression for this particular term, using a [JavaScript-flavored regular expression string](#).
 - `caseSensitive` (Boolean) Determines whether we consider case when matching this term. Default is `false`.
 - `distance` (Integer) **Required.** Maximum number of words allowed between any two consecutive sub-terms.
 - `contextPadding` (Integer) Maximum number of characters to include both before and after the search result in the returned `context` string. For example, a value of 25 would allow up to 25 preceding and 25 following characters of content. Default is 25.
 - `termId` (String) Optional id of your choosing which, if provided, will be included as a `termId` property on each search result produced by this term. When used, we do not enforce uniqueness; it is your responsibility to use a unique `termId` for each term.
- `minSecondsAvailable` (Integer) The minimum number of seconds this search task will remain available. The actual lifetime may be longer. The default lifetime is defined by the `processIds.lifetime` [central configuration](#) parameter.

Successful Response

Response Body

JSON with metadata about the created search task.

- `input` (Object) Input we used to create the search task. May include default values not explicitly provided in the original POST.
- `processId` (String) Unique id for this search task.
- `affinityToken` (String) Affinity token for this search task. Present when clustering is enabled.
- `state` (String) State of getting search results.
 - "processing" - The search is still being executed. Additional results may become available.
 - "complete" - The search is complete. No additional results will become available.
 - "error" - There was a problem performing the search. No additional results will become available.
- `percentComplete` (Integer) Percentage of the document text which has been searched (from 0 to 100).
- `expirationDateTime` (String) Currently planned date and time when the search task resource will expire and no longer be available for use. Format is [RFC 3339 Internet Date/Time profile of ISO 8601](#), e.g. "2016-11-05T08:15:30.494Z".

Error Responses

Status Code	JSON errorCode	Description
404	-	No viewing session with the provided {viewingSessionId} could be found.
480	"DocumentNotProvidedYet"	The viewing session does not yet have a source document attached.
480	"MissingInput"	A required input value was not provided. See <code>errorDetails</code> in the response body.
480	"InvalidInput"	An invalid input value was used. See <code>errorDetails</code> in the response body.
480	"MissingInputForSimpleTerm"	An invalid input value was used in a "simple" term object. See <code>errorDetails</code> in the response body.
480	"InvalidInputForSimpleTerm"	An invalid input value was used in a "simple" term object. See <code>errorDetails</code> in the response body.
480	"MissingInputForProximityTerm"	An invalid input value was used in a "proximity" term object. See <code>errorDetails</code> in the response body.
480	"InvalidInputForProximityTerm"	An invalid input value was used in a "proximity" term object. See <code>errorDetails</code> in the response body.
480	"FeatureDisabled"	The viewing session was created with "serverSideSearch" disabled.
580	"InternalError"	The server encountered an internal error when handling the request.

Example

Request

```
POST
prizmdoc_server_base_url/v2/viewingSessions/DLbVh9sTmXJAmldGeXbS9Gn3WHxs8oib2xPsW2xEFjnIDdoJcudPtxciodsYFQq6zYGabQ_rJIecdbkImTTkSA/searchTasks
Content-Type: application/json
```

```
{
  "input": {
    "searchTerms": [{
      "type": "simple",
      "pattern": "quick"
    }]
  }
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "searchTerms": [{
      "type": "simple",
      "pattern": "quick",
      "caseSensitive": false,
      "contextPadding": 25
    }]
  },
  "processId": "pR5X6nPDgMwat6cx1mn0Q3",
  "state": "processing",
  "percentComplete": 0,
  "expirationDateTime": "2016-12-17T20:38:39.796Z"
}
```

Additional Examples

For more examples of how to construct different searches, see [Example Searches](#).

GET /v2/searchTasks/{processId}

Gets information about a search task.

To get search results, use [GET /v2/searchTasks/{processId}/results](#).

Request

URL Parameters

Parameter	Description
{processId}	The <code>processId</code> which identifies the search task.

Request Headers

Name	Description
Accusoft-Affinity-Token	The <code>affinityToken</code> of the search task. Required when server clustering is enabled.

Successful Response

Response Body

JSON with metadata about the search task.

- `input` (Object) Input we used to create the search task. May include default values not explicitly provided in the original POST.
- `processId` (String) Unique id for this search task.
- `affinityToken` (String) Affinity token for this search task. Present when clustering is enabled.
- `state` (String) State of getting search results.
 - "processing" - The search is still being executed. Additional results may become available.
 - "complete" - The search is complete. No additional results will become available.
 - "error" - There was a problem performing the search. No additional results will become available.
- `percentComplete` (Integer) Percentage of the document text which has been searched (from 0 to 100).
- `expirationDateTime` (String) Currently planned date and time when the search task resource will expire and no longer be available for use. Format is [RFC 3339 Internet Date/Time profile of ISO 8601](#), e.g. "2016-11-05T08:15:30.494Z".

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No search task with the provided <code>{processId}</code> could be found.
400	"MissingInput"	Can occur when clustering is enabled and an <code>Accusoft-Affinity-Token</code> request header was not provided.
500	"InternalError"	The server encountered an internal error when handling the request.

Example

Request

```
GET prizmdoc\_server\_base\_url/v2/searchTasks/pR5X6nPDgMwat6cx1mn0Q3
Accusoft-Affinity-Token: eJN9/kXEYouken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "contextId": "ElkNzWtrUJp4rXI5YnLUgw",
    "searchTerms": [{
      "type": "simple",
      "pattern": "quick",
      "caseSensitive": false,
      "contextPadding": 25
    }]
  },
  "processId": "pR5X6nPDgMwat6cxlmn0Q3",
  "state": "complete",
  "percentComplete": 100,
  "expirationDateTime": "2016-12-17T20:38:39.796Z"
}
```

GET /v2/searchTasks/{processId}/results?limit={limit}&continueToken={continueToken}

Gets a block of newly-available search results up to a limit.

This URL is designed to give you the results in chunks as they become available. Each GET request will return the currently-known results up to a `limit` (default is 100). If a response contains a `continueToken`, it indicates that additional results may be available and that you should issue another GET request using that `continueToken` as a query string parameter to skip the results you have already received. As long as a response contains a `continueToken`, use it to issue a subsequent GET for more results. When you encounter a response which does *not* have a `continueToken`, you have received all of the results and no more GET requests are necessary.

In order to optimize the number of network requests you make, any response which contains a `continueToken` will also contain a `continueAfter` value with a recommended number of milliseconds you should wait before sending the next GET request.

Request

URL Parameters

Parameter	Description
{processId}	The <code>processId</code> which identifies the search task.
{limit}	The maximum number of results to return for this HTTP request. Must be an integer greater than 0. Default is 100.
{continueToken}	Used to continue getting results from the point where a previous GET request left off.

Request Headers

Name	Description
Accusoft-Affinity-Token	The <code>affinityToken</code> of the search task. Required when server clustering is enabled.

Successful Response

Response Body

JSON with any available search results.

- **results** (Array of Objects) **Always present.** Array of newly-available search results. If no new results are available, this array will be empty.
 - `id` (Integer) Unique number assigned to this search result.
 - `pageIndex` (Integer) Zero-indexed page number where this search result occurs in the document.
 - `text` (String) Text which was matched.
 - `context` (String) Contextual excerpt, including the matched text itself. The amount of leading and trailing characters to include in this value is controlled by `input.contextPadding` in the initial POST to create the search task.
 - `boundingRectangle` (Object) Bounding rectangle dimensions of the matched text on the page where it occurs.
 - `x` (Number) Distance from the left edge of the page to the left edge of the search result bounding box.
 - `y` (Number) Distance from the top edge of the page to the top edge of the search result bounding box.
 - `width` (Number) Width of the search result bounding box.
 - `height` (Number) Height of the search result bounding box.
 - `lineRectangles` (Array of Objects) Array of rectangles for each line of the matched text on the page where it occurs. If the match is on one line, the result is a single array item with a `rectangle` equal to `boundingRectangle`. If the match is on multiple lines, all rectangles in the array will be within the bounds of the `boundingRectangle`.
 - `x` (Number) Distance from the left edge of the page to the left edge of the search result line rectangle.
 - `y` (Number) Distance from the top edge of the page to the top edge of the search result line rectangle.
 - `width` (Number) Width of the search result line rectangle.
 - `height` (Number) Height of the search result line rectangle.
 - `pageData` (Object) Information about the dimensions of the page where this search result occurs.
 - `width` (Number) Width of the page.
 - `height` (Number) Height of the page.
 - `searchTerm` (Object) Search term which produced this result. The value will correspond to one of the items passed in to `input.searchTerms` in the initial POST to create the search task.
 - **When `type` is "simple":**
 - `type` (String) **Always present** with a value of "simple".
 - `pattern` (String) **Always present.** Regular expression which produced this result.
 - `caseSensitive` (Boolean) **Always present.** Indicates whether or not case was considered for this result.
 - `contextPadding` (Integer) **Always present.** Amount of context padding requested for this term in the initial POST.
 - `termId` (String) When provided in the initial POST, `termId` of the term which produced this result.
 - **When `type` is "proximity":**
 - `type` (String) **Always present** with a value of "proximity".
 - `subTerms[]` (Array of Objects) **Always present.** The sub-terms which contributed to this result. Each item will contain:

- `pattern` (String) **Always present.** Regular expression for this particular sub-term.
 - `caseSensitive` (Boolean) **Always present.** Indicates whether or not case was considered when matching this particular sub-term in the result.
 - `distance` (Integer) **Always present.** Maximum number of words allowed between any two consecutive sub-terms.
 - `contextPadding` (Integer) **Always present.** Amount of context padding requested for this term in the initial POST.
 - `termId` (String) When provided in the initial POST, `termId` of the term which produced this result.
- `startIndex` (Integer) JavaScript string index into the full-page text string where the matched text begins (to get the full-page text string, see [GET /v2/searchContexts/{contextId}/records](#)).
- `startIndexInContext` (Integer) JavaScript string index into the returned `context` string where the matched text begins.
- `pagesWithoutText` (Array of Integers) **Always present.** Currently known pages in the document which do not contain any text content at all. Values are zero-indexed page numbers. If the search task is still processing (a `continueToken` is present in the response), the data should be considered partial. Note that, unlike `results`, this value is cumulative (we always deliver the entire set of pages we know to not contain text data).
- `continueToken` (String) When present, indicates that more search results may be available. An additional GET request should be made for more results using this value as the `continueToken` query string parameter. When not present, indicates that the search is complete and no further results will be available.
- `continueAfter` (Number) Recommended milliseconds to delay before issuing the next GET request for more results.

Error Responses

Status Code	JSON errorCode	Description
404	-	No search task with the provided <code>{processId}</code> could be found.
400	"MissingInput"	Can occur when clustering is enabled and an <code>Accusoft-Affinity-Token</code> request header was not provided.
480	"InvalidInput"	An invalid input value was used. See <code>errorDetails</code> in the response body.
480	"ResourceNotUsable"	Can occur when the search task is in a state of "error". You may be able to get more information from a <code>GET /v2/searchTasks/{processId}</code> .
580	"InternalError"	The server encountered an internal error when handling the request.

Example

Say you have a search task which was created to find the regex `"manag[a-z]*"` in a particular whitepaper. Here is an example sequence of requests and responses illustrating how you would acquire the full set of results for the search task (for brevity, the total number of search results in this example is small).

You would start with an initial GET:

```
GET prizmdoc\_server\_base\_url/v2/searchTasks/pR5X6nPDgMwat6cxlmn0Q3/results
Accusoft-Affinity-Token: eJN9/kXEYOuken4Pb9ic9hqJk45XIad9LQNgCgQ+BkM=
```

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "results": [
    {
      "id": 0,
      "pageIndex": 0,
      "text": "Management",
      "context": "Enterprise Content Management Best Practices",
      "boundingRectangle": { "x": 24.20, "y": 13.74, "width": 234.20, "height": 26.10 },
      "lineRectangles": [ { "x": 24.20, "y": 13.74, "width": 234.20, "height": 26.10 } ],
      "pageData": { "width": 612, "height": 792 },
      "searchTerm": {
        "type": "simple",
        "pattern": "manag[a-z]*",
        "caseSensitive": false,
        "contextPadding": 25
      },
      "startIndex": 19,
      "startIndexInContext": 19
    },
    {
      "id": 1,
      "pageIndex": 0,
      "text": "management",
      "context": "ue of enterprise content management software should go way b",
      "boundingRectangle": { "x": 156.07, "y": 352.19, "width": 105.00, "height": 13.41 },
      "lineRectangles": [ { "x": 156.07, "y": 352.19, "width": 105.00, "height": 13.41 } ],
      "pageData": { "width": 612, "height": 792 },
      "searchTerm": {
        "type": "simple",
        "pattern": "manag[a-z]*",
        "caseSensitive": false,
        "contextPadding": 25
      },
      "startIndex": 527,
      "startIndexInContext": 25
    }
  ],
  "pagesWithoutText": [],
  "continueToken": "Cx07GH1kmi32gxAQhv49WZ",
  "continueAfter": 500
}
```

The initial response has given us two results for the first page of the document (page index 0) and a `continueToken` which we should use to get more results after waiting 500 milliseconds.

So, half a second later, we issue a follow-up request with the `continueToken` passed in as a query string parameter (so we skip over the results we already have):

```
GET prizmdoc\_server\_base\_url/v2/searchTasks/pR5X6nPDgMwat6cxlmn0Q3/results?continueToken=Cx07GH1kmi32gxAQhv49WZ
Accusoft-Affinity-Token: eJN9/kXEYOuken4Pb9ic9hqJk45XIad9LQNgCgQ+BkM=
```

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json

{
  "results": [
    {
      "id": 2,
      "pageIndex": 1,
      "text": "management",
      "context": "Enterprise content management software helps eliminate",
      "boundingRectangle": { "x": 310.21, "y": 562.14, "width": 254.03, "height": 26.10 },
      "lineRectangles": [{ "x": 310.21, "y": 562.14, "width": 254.03, "height": 26.10 }],
      "pageData": { "width": 612, "height": 792 },
      "searchTerm": {
        "type": "simple",
        "pattern": "manag[a-z]*",
        "caseSensitive": false,
        "contextPadding": 25
      },
      "startIndex": 652,
      "startIndexInContext": 19
    }
  ],
  "pagesWithoutText": [2,3],
  "continueToken": "B4uGe7m0ZtxR3lkqA07Nmj",
  "continueAfter": 500
}
```

This time we get back a new result as well as some new information about `pagesWithoutText`: we now know that at least page indices 2 and 3 (zero-indexed page numbers) have no text at all. The presence of a new `continueToken` tells us there may be more results, so we submit another request with the new `continueToken`:

```
GET /prizmdoc_server_base_url/v2/searchTasks/pr5X6nPDgMwat6cxlmn0Q3/results?continueToken=B4uGe7m0ZtxR3lkqA07Nmj
Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "results": [
    {
      "id": 3,
      "pageIndex": 5,
      "text": "management",
      "context": "upply chains to contract management, or HR processes to gove",
      "boundingRectangle": { "x": 67.00, "y": 142.53, "width": 254.03, "height": 26.10 },
      "lineRectangles": [{ "x": 67.00, "y": 142.53, "width": 254.03, "height": 26.10 }],
      "pageData": { "width": 612, "height": 792 },
      "searchTerm": {
        "type": "simple",
        "pattern": "manag[a-z]*",
        "caseSensitive": false,
        "contextPadding": 25
      },
      "startIndex": 113,
      "startIndexInContext": 25
    }
  ],
  "pagesWithoutText": [2,3,4]
}
```

This time we get a new result for page index 5, and we now know that page indices 2, 3, and 4 all contain no text at all (apparently this was not much of a whitepaper!). The lack of a `continueToken` tells us we have received all of the results, so there are no more GET requests to make.

DELETE /v2/searchTasks/processId

Cancels the search task. Further requests using this `processId` will return errors.

Request

URL Parameters

Parameter	Description
{processId}	The <code>processId</code> which identifies the search task.

Request Headers

Name	Description
Accusoft-Affinity-Token	The <code>affinityToken</code> of the search task. Required when server clustering is enabled.

Successful Response

```
HTTP/1.1 204 No Content
```

Error Responses

Status Code	JSON errorCode	Description
404	-	No search task with the provided {processId} could be found.
400	"MissingInput"	Can occur when clustering is enabled and an Accusoft-Affinity-Token was not provided.
580	"InternalError"	The server encountered an internal error when handling the request.

Example Searches

The following examples demonstrate how to use `input.searchTerms` for both the POST `/v2/searchTasks` and POST `/v2/viewingSessions/{viewingSessionId}/searchTasks` URLs.

Start a search for a single word

This partial input JSON begins a search task which finds all instances of the word "quick":

```
{
  "input": {
    "searchTerms": [{
      "type": "simple",
      "pattern": "quick"
    }]
  }
}
```

Start a case-sensitive search for an exact phrase

This partial input JSON begins a case-sensitive search for the exact phrase "The quick brown fox jumped over the lazy dog.". Notice that we had to escape the period character because it is a special regex character (`\.`), and because this is a JSON string value, the backslash itself must also be escaped (`\\.`):

```
{
  "input": {
    "searchTerms": [{
      "type": "simple",
      "pattern": "The quick brown fox jumped over the lazy dog\\.",
      "caseSensitive": true
    }]
  }
}
```

Start a search for every instance of the word "quick" or "brown" or "fox"

This partial input JSON begins a search for the words "quick" or "brown" or "fox", locating all instances of each of these words:

```
{
  "input": {
    "searchTerms": [
      {
        "type": "simple",
        "pattern": "quick"
      },
      {
        "type": "simple",
        "pattern": "fox"
      },
      {
        "type": "simple",
        "pattern": "dog"
      }
    ]
  }
}
```

Start a search for "quick" and "fox" and "dog" where there are no more than 5 words between any two consecutive occurrences of them

```
{
  "input": {
    "searchTerms": [
      {
        "type": "proximity",
        "subTerms": [
          {
            "pattern": "quick"
          },
          {
            "pattern": "fox"
          },
          {
            "pattern": "dog"
          }
        ],
        "distance": 5
      }
    ]
  }
}
```

Start a case-sensitive search for "John Doe" within 30 words of what looks like a social security number

```
{
  "input": {
    "searchTerms": [
      {
        "type": "proximity",
        "subTerms": [
          {
            "pattern": "John Doe",
            "caseSensitive": true
          }
        ]
      }
    ]
  }
}
```



```

    }, {
      "pattern": "\\d{3}-\\d{2}-\\d{4}"
    }],
    "distance": 30
  }]
}
}

```

Work Files

Introduction

The *work file* REST API provides a temporary storage system to upload file input and download file output. Each work file has a unique `fileId` which can be used to pass that file as input to a process or viewing session or to download the raw bytes of the file.

Work files should not be used for archival storage. All work files are temporary and, by default, will be deleted 24 hours after creation.

Available URLs

URL	Description
POST /PCCIS/V1/WorkFile when body is file bytes	Creates a work file resource from file bytes (used to upload input files).
POST /PCCIS/V1/WorkFile when body is JSON	Creates a "package" work file from a set of existing work files. Used to prepare a set of dependent files for viewing or processing (e.g. CAD with XREF).
GET /PCCIS/V1/WorkFile/{fileId}	Gets the file bytes associated with a work file resource (used to download output files).
GET /PCCIS/V1/WorkFile/{fileId}/Info	Gets metadata information about a work file resource.

POST /PCCIS/V1/WorkFile when body is file bytes

Creates a work file resource from file bytes (used to upload input files).

Example

```

POST prizmdoc\_server\_base\_url/PCCIS/V1/WorkFile
Content-Type: application/octet-stream

<<file bytes>>

```

```

HTTP/1.1 200 OK
Content-Type: application/json

{
  "fileId": "Xe6zv3dH0kVSzLuaNhd32A",
  "fileExtension": "pdf",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}

```

Request

Request Headers

Name	Description
Content-Type	We recommend you use the value <code>application/octet-stream</code> to explicitly indicate that you are uploading file bytes. If you do not provide a value, <code>application/octet-stream</code> is assumed by default.
Accusoft-Affinity-Token	Used to ensure that this work file will be assigned to the same machine in the cluster as another existing resource (work file, process, or viewing session). If provided, the value must be the <code>affinityToken</code> of another existing resource. If not provided, an <code>affinityToken</code> will be randomly assigned to each work file created whenever there is more than one server in the cluster.

Request Parameters

```
POST prizmdoc_server_base_url/PCCIS/V1/WorkFile{?
FileExtension,MinSecondsAvailable}
```

Name	Type	Description	Example
FileExtension	string	File extension of the file being uploaded without any leading period. Only required when the file type cannot be automatically detected (e.g. <code>csv</code> , <code>tsv</code> , <code>txt</code> , <code>eml</code>). The <code>FileExtension</code> parameter may only accept alpha-numeric characters. Note that, if we are able to detect the file type automatically, the detected type will always be used and this value will be ignored.	<code>pdf</code>
MinSecondsAvailable	integer	The minimum number of seconds this work file must remain available. The actual lifetime may be longer.	<code>300</code>

Request Body

The bytes of the file being uploaded.

Successful Response

JSON with metadata about the newly created work file.

Response Headers

Name	Value
Content-Type	<code>application/json</code>

Response Body

```
{
  "fileId": "Xe6zv3dH0kVSzLuaNhd32A",
```

```

"fileExtension": "pdf",
"affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}

```

Name	Type	Description
fileId	string	The unique id of the new work file resource, often used later as input to a viewing session or process.
fileExtension	string	The file extension assigned to this resource which indicates what type of file we understand it to be.
affinityToken	string	A value used to identify which machine in the cluster this work file resides on. Only present when there is more than one machine in the cluster. A work file is only accessible to another resource if that resource resides on the same machine in the cluster. You can ensure that other resources you create (work files, viewing sessions, and processes) are assigned to the same machine by passing this value in an <code>Accusoft-Affinity-Token</code> request header when submitting the POST to create the other resource.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
400	-	There was a problem with an input parameter: <ul style="list-style-type: none"> <code>FileExtension</code> was required and not provided. <code>FileExtension</code> was an invalid value. <code>MinSecondsAvailable</code> was not a number.
405	-	An HTTP method other than POST was used.
580	"UnrecognizedFileFormat"	The file type of the uploaded bytes could not be automatically detected. You will need to manually specify a <code>FileExtension</code> .

POST /PCCIS/V1/WorkFile when body is JSON

Creates a "package" work file from a set of existing work files, defining one file in the set as the primary file for viewing and content conversion.

This special type of work file is particularly useful for CAD drawings which are made up of multiple files (such as dwg files which use other files via XREF).

Example

```

POST prizmdoc\_server\_base\_url/PCCIS/V1/WorkFile
Content-Type: application/json
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

{
  "primaryPath": "master.dwg",

```

```

"items": [
  { "fileId": "CVBuD7DbQYNoJDqByGierQ", "path": "master.dwg" },
  { "fileId": "5qTYa3gzN9gYUb5SzqUhgq", "path": "parts/a.dwg" },
  { "fileId": "o1bLJwFGxf9QGutkyrOqig", "path": "parts/b.dwg" }
]
}

```

```

HTTP/1.1 200 OK
Content-Type: application/json

{
  "fileId": "nkG9fiAmj27X3MhqGdbsXA",
  "primaryPath": "master.dwg",
  "items": [
    { "fileId": "CVBuD7DbQYNoJDqByGierQ", "path": "master.dwg" },
    { "fileId": "5qTYa3gzN9gYUb5SzqUhgq", "path": "parts/a.dwg" },
    { "fileId": "o1bLJwFGxf9QGutkyrOqig", "path": "parts/b.dwg" }
  ],
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}

```

Request

Request Headers

Name	Description
Content-Type	You must use the value <code>application/json</code> to indicate that you are assembling a new work file from a set of existing work files rather than simply uploading file bytes. If you forget to specify a <code>Content-Type</code> of <code>application/json</code> , we will assume you are attempting to upload file bytes.
Accusoft-Affinity-Token	Whenever there is more than one machine in the cluster, you will need to ensure that all of the work files in the set reside on to the same machine. To do this, upload the first file in the set, get the <code>affinityToken</code> in the response, and then use that value in an <code>Accusoft-Affinity-Token</code> request header for every subsequent work file POST, including this final POST to assemble a work file for the entire set.

Request Parameters

```
POST prizmdoc_server_base_url/PCCIS/V1/WorkFile{?MinSecondsAvailable}
```

Name	Type	Description	Example
MinSecondsAvailable	integer	The minimum number of seconds this work file must remain available. The actual lifetime may be longer.	300

Request Body

A JSON object which specifies:

- which work files are in the set
- what their local paths would be

- which path should be considered the primary entry point for viewing or conversion

For example:

```
{
  "primaryPath": "master.dwg",
  "items": [
    { "fileId": "CVBuD7DbQYNoJDqByGierQ", "path": "master.dwg" },
    { "fileId": "5qTYa3gzN9gYUb5SzqUhqg", "path": "parts/a.dwg" },
    { "fileId": "o1bLJwFGxf9QGuTkyrOqig", "path": "parts/b.dwg" }
  ]
}
```

Name	Type	Description
<code>primaryPath</code>	string	Required. The primary entry point for viewing or conversion. This value must match the <code>path</code> value of one of the objects in the <code>items</code> array.
<code>items</code>	array	Required. The work files which are to be included in the set and what their local paths would be.

Items

Name	Type	Description
<code>fileId</code>	string	Required. The <code>fileId</code> of a work file to be included in the set.
<code>path</code>	string	Required. A path value for this file in the set. Typically this is just the relative path of this file in relation to the primary file. Each item in the set must have a unique <code>path</code> value.

Successful Response

JSON with metadata about the new work file resource, the most important part being a new `fileId` which you can use to represent the entire set of files.

Response Headers

Name	Value
<code>Content-Type</code>	<code>application/json</code>

Response Body

```
{
  "fileId": "nkG9fiAmj27X3MhqGdbsXA",
  "primaryPath": "master.dwg",
  "items": [
    { "fileId": "CVBuD7DbQYNoJDqByGierQ", "path": "master.dwg" },
    { "fileId": "5qTYa3gzN9gYUb5SzqUhqg", "path": "parts/a.dwg" },
    { "fileId": "o1bLJwFGxf9QGuTkyrOqig", "path": "parts/b.dwg" }
  ],
  "affinityToken": "ejN9/kXEYouken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

Name	Type	Description
<code>fileId</code>	string	The unique id of this new "package" work file, intended to be used as input to a viewing session or process which needs to consume the entire package as a single document.
<code>primaryPath</code>	string	The primary entry point which will be used for viewing or conversion.
<code>items</code>	array	The work files which are included in the set and their assigned paths.
<code>affinityToken</code>	string	A value used to identify which machine in the cluster this work file resides on. Only present when there is more than one machine in the cluster. A work file is only accessible to another resource if that resource resides on the same machine in the cluster. You can ensure that other resources you create (work files, viewing sessions, and processes) are assigned to the same machine by passing this value in an <code>Accusoft-Affinity-Token</code> request header when submitting the POST to create the other resource.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
480	"MissingInput"	A required input was not provided. See the <code>errorDetails</code> in the response.
480	"InvalidInput"	One of the input values was invalid. Possible causes: <ul style="list-style-type: none"> <code>MinSecondsAvailable</code> was not a number. <code>primaryPath</code> did not match the <code>path</code> of any of the <code>items</code>. See the <code>errorDetails</code> in the response.
480	"ValueMustBeUnique"	One or more of the <code>path</code> values in the <code>items</code> array was non-unique.
580	UnrecognizedFileFormat	This will occur if you forget to specify a <code>Content-Type</code> of <code>application/json</code> when making your request. If no <code>Content-Type</code> is specified (or if any value other than <code>application/json</code> is specified), we will understand you to be making a request to upload file bytes rather than giving us JSON instructions to create a new work file from a set of existing work files. And, since we do not automatically detect JSON as a file format, this is the error which is returned. If you are trying to create a work file from a set of existing work files (as described in this section), make sure you set the request <code>Content-Type</code> header to <code>application/json</code> .

GET /PCCIS/V1/WorkFile/{fileId}

Gets the file bytes associated with a work file resource (used to download output files).

Example

Get work file with default filename

```
GET prizmdoc\_server\_base\_url/PCCIS/V1/WorkFile/Xe6zv3dH0kVSzLuaNhd32A
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

```
HTTP/1.1 200 OK
Content-Type: application/pdf
Content-Disposition: attachment; filename="file-Xe6zv3dH0kVSzLuaNhd32A.pdf"

<<file bytes>>
```

Get work file with non-ASCII character in the filename

```
GET prizmdoc\_server\_base\_url/PCCIS/V1/WorkFile/Xe6zv3dH0kVSzLuaNhd32A?
ContentDispositionFilename=GreekABΓΔ
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

```
HTTP/1.1 200 OK
Content-Type: application/pdf
Content-Disposition: attachment; filename="file-Xe6zv3dH0kVSzLuaNhd32A.pdf";
filename*=UTF-8' 'Greek%CE%91%CE%92%CE%93%CE%94.pdf`

<<file bytes>>
```

Request

Request Headers

Name	Description
Accusoft-Affinity-Token	If there is more than one machine in the cluster (an <code>affinityToken</code> was provided in the response to the initial POST request to create the work file), then this header must be set to the <code>affinityToken</code> of the work file you are trying to access.

Request Parameters

```
GET prizmdoc\_server\_base\_url/PCCIS/V1/WorkFile/{fileId}?
ContentDispositionFilename}
```

Name	Type	Description	Example
<code>fileId</code>	string	Required. The <code>fileId</code> of the work file whose file bytes you want to download.	<code>Xe6zv3dH0kVSzLuaNhd32A</code>
<code>ContentDispositionFilename</code>	string	The filename as a URL-encoded string, without extension, to be used in the <code>Content-</code>	<code>GreekABΓΔ</code>

Name	Type	Description	Example
		<code>Disposition</code> response header (the file extension will be appended automatically). The default is <code>file-<fileId></code> .	

Successful Response

The raw bytes of the file.

Response Headers

Name	Description	Example
<code>Content-Type</code>	The MIME type of the file being downloaded.	<code>application/pdf</code>
<code>Content-Disposition</code>	Specifies 'attachment' disposition, RFC-2183 compatible <code>filename</code> parameter and, if <code>ContentDispositionFilename</code> was specified in the request, RFC-8187 compatible <code>filename*</code> parameter, allowing the use of non-ASCII filenames.	<code>attachment;</code> <code>filename="Greek____.pdf";</code> <code>filename*=UTF-</code> <code>8' 'Greek%CE%91%CE%92%CE%93%CE%94.pdf</code>

Response Body

The raw bytes of the file.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
400	"InvalidInput"	One of the input values was invalid. Possible causes: <ul style="list-style-type: none"> <code>Accusoft-Affinity-Token</code> was invalid. See the <code>errorDetails</code> in the response.
480	"DataNotAvailable"	No file bytes exist for download. This can occur if you attempt to download the file bytes for a ["package" work file] (#post-json) which merely groups a set of related work files under a single <code>fileId</code> .
404	-	No work file existed for the given <code>fileId</code> .
405	-	An HTTP method other than GET was used.

GET /PCCIS/V1/WorkFile/{fileId}/Info

Gets metadata information about a work file resource, the same information returned in the original POST request which created the work file.

Example


```
GET prizmdoc_server_base_url/PCCIS/V1/WorkFile/Xe6zv3dH0kVSzLuaNhd32A/Info
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "fileId": "Xe6zv3dH0kVSzLuaNhd32A",
  "fileExtension": "pdf",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

Request

Request Headers

Name	Description
Accusoft-Affinity-Token	If there is more than one machine in the cluster (an <code>affinityToken</code> was provided in the response to the initial POST request to create the work file), then this header must be set to the <code>affinityToken</code> of the work file you are trying to access.

Request Parameters

```
GET prizmdoc_server_base_url/PCCIS/V1/WorkFile/{fileId}/Info
```

Name	Type	Description	Example
<code>fileId</code>	string	Required. The <code>fileId</code> of the work file you need information about.	Xe6zv3dH0kVSzLuaNhd32A

Successful Response

JSON with information about the work file.

Response Headers

Name	Value
Content-Type	application/json

Response Body

For simple work files

```
{
  "fileId": "Xe6zv3dH0kVSzLuaNhd32A",
  "fileExtension": "pdf",
}
```

```

    "affinityToken": "ejN9/kXEYouken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
  }

```

Name	Type	Description
<code>fileId</code>	string	The unique id of this resource.
<code>fileExtension</code>	string	File extension assigned to this resource, indicating what type of file we understand it to be.
<code>affinityToken</code>	string	A value used to identify which machine in the cluster this work file resides on. Only present when there is more than one machine in the cluster.

For "package" work files

```

{
  "fileId": "nkG9fiAmj27X3MhqGdbsXA",
  "primaryPath": "master.dwg",
  "items": [
    { "fileId": "CVBuD7DbQYNoJDqByGierQ", "path": "master.dwg" },
    { "fileId": "5qTYa3gzN9gYUb5SzqUhgq", "path": "parts/a.dwg" },
    { "fileId": "olbLJwFGxf9QGuTkyrOqig", "path": "parts/b.dwg" }
  ],
  "affinityToken": "ejN9/kXEYouken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}

```

Name	Type	Description
<code>fileId</code>	string	The unique id of this resource.
<code>primaryPath</code>	string	The primary entry point which will be used for viewing or conversion.
<code>items</code>	array	The work files which are included in the set and their assigned paths.
<code>affinityToken</code>	string	A value used to identify which machine in the cluster this work file resides on. Only present when there is more than one machine in the cluster.

Error Responses

Status Code	Description
404	No work file existed for the given <code>fileId</code> .
405	An HTTP method other than GET was used.

Self-Hosted Administration

These REST APIs are useful if you are self-hosting PrizmDoc Server instances:

- [Health Status](#)
- [Cluster Management](#)

Health Status

Introduction

For customers who are self-hosting PrizmDoc Server, the *health* REST API allows an administrator or application to check the health of a PrizmDoc Server instance.

Available URLs

URL	Description
GET /PCCIS/V1/Service/Current/Health	Determines whether or not a specific PrizmDoc Server instance is healthy.
GET /PCCIS/V1/Service/Current/Info	Gets metadata about the current status of a specific PrizmDoc Server instance.

GET /PCCIS/V1/Service/Current/Health

Determines whether or not a specific PrizmDoc Server instance is healthy.

Response

You can use the HTTP status code to determine the health of the instance:

HTTP Status Code	Status
200	Healthy
500	Unhealthy

You can also determine health by looking at the response body. When healthy, the response body will be plaintext with a value of `OK`. When unhealthy, the response will have no body.

NOTE: *If PrizmDoc Viewer has just started, unhealthy status may be returned for a short time until the system has completely started up.*

Example

Request

```
GET http://prizmdoc_server_base_url/PCCIS/V1/Service/Current/Health
```

Response

```
HTTP/1.1 200 OK  
OK
```

GET /PCCIS/V1/Service/Current/Info

Gets metadata about the current status of a specific PrizmDoc Server instance.

NOTE: Do not take any actions when any child service reports `unhealthy` status. PrizmDoc Server has its own mechanism to recover.

Successful Response

Response Body

JSON with the following:

- `serviceStatus` (String) Status of PrizmDoc Server:
 - "starting"
 - "running"
 - "unhealthy"
- `licenseStatus` (String) Information about the PrizmDoc Server License:
 - "retrieving" - Returned when the product is starting and the licensed status is not yet known.
 - "unlicensed" - Returned when the provided license key is not valid. Your license may be expired, or you may not have correctly configured your license key.
 - "licensed as 'Your Solution Name'" - Returned when a valid license key is found. *Your Solution Name* will be replaced with the value of your actual solution name.
- `instances[]` (Array of Objects) Objects describing the PrizmDoc Server instances that are currently running. Items contain:
 - `serviceStatus` (String) Status of the PrizmDoc Server used for viewing.
 - `serviceInstallerVersion` (String) Version of the installer used to install PrizmDoc Server.
 - `pccisVersion` (String) Version of the PrizmDoc Server used for viewing.
 - `runtimeVersion` (String) .NET runtime version supporting the PrizmDoc Server used for viewing.
 - `operatingSystem` (String) Operating System of the server on which PrizmDoc Server is running.
 - `startTime` (String) Last recorded time the PrizmDoc Server were started. Time is reported in UTC and is ISO-8601 format.
 - `instanceId` (String) Host name of the server running PrizmDoc Server.
 - `childServices[]` (Array of Objects) Objects describing the health status for each individual PrizmDoc Service. > **NOTE:** It's important not to programmatically depend upon the value of a Service name because the total number of items in the `childServices` array may change from release to release (items may be added or removed). Items may contain:
 - `name` (String) Name of the child PrizmDoc Service.
 - `status` (String) Status of the child PrizmDoc Service:
 - "starting"
 - "running"
 - "unhealthy"

Example

Request

```
GET http://prizmdoc_server_base_url/PCCIS/V1/Service/Current/Info
```

Response

```
HTTP/1.1 200 OK
```

Content-Type: application/json

```
{
  "serviceStatus": "running",
  "licenseStatus": "licensed as 'Acme Inc.'",
  "instances": [
    {
      "serviceStatus": "running",
      "serviceInstallerVersion": "XX.X.XX.XXX",
      "pccisVersion": "XX.X.XX.XXXX",
      "runtimeVersion": "4.0.30319.34014",
      "operatingSystem": "Microsoft Windows NT 6.3.9600.0",
      "startTime": "1971-01-01T00:00:00.0Z",
      "instanceId": "myhostname",
      "childServices": [
        {
          "name": "PCC Error Reporting Service",
          "serviceStatus": "running"
        },
        {
          "name": "PCC Imaging Conversion Service",
          "serviceStatus": "running",
          "version": "X.X.XXXX.XXXX"
        },
        {
          "name": "PCC PDF Processing Service",
          "serviceStatus": "running"
        },
        {
          "name": "PCC Raster Conversion Service",
          "serviceStatus": "running",
          "version": "X.X.XXXX.XXXX"
        },
        {
          "name": "PCC Vector Conversion Service",
          "serviceStatus": "running",
          "version": "X.X.XXXX.XXXX"
        },
        {
          "name": "PCC Html Conversion Service",
          "serviceStatus": "running",
          "version": "X.X.XXXX.XXXX"
        },
        {
          "name": "PCC Work File Service",
          "serviceStatus": "running",
          "version": "X.X.X"
        },
        {
          "name": "PCC Office Conversion Service",
          "serviceStatus": "running",
          "version": "X.XX.XXXX.XXXX"
        },
        {
          "name": "PCC Format Detection Service",
          "serviceStatus": "running"
        },
        {
          "name": "PCC AutoRedaction Service",
          "serviceStatus": "running"
        }
      ]
    }
  ]
}
```

```
    "name": "PCC Redaction Service",
    "serviceStatus": "running",
    "version": "X.X.X"
  },
  {
    "name": "PCC Email Processing Service",
    "serviceStatus": "running"
  },
  {
    "name": "PCC Email Conversion Service",
    "serviceStatus": "running",
    "version": "X.X.XXXX.XXXX"
  },
  {
    "name": "PCC Content Conversion Service",
    "serviceStatus": "running"
  },
  {
    "name": "configuration-service",
    "serviceStatus": "running"
  },
  {
    "name": "licensing-service",
    "serviceStatus": "running"
  },
  {
    "name": "health-service",
    "serviceStatus": "running"
  }
]
}
```

Cluster Management

Introduction

For customers self-hosting PrizmDoc Server, the *cluster management* REST API allows you to inform a PrizmDoc Server instance whenever a new instance is added to or removed from the cluster.

NOTE: The cluster management REST API is only available if 1) you are self-hosting PrizmDoc Server and 2) you have clustering enabled. If you are using PrizmDoc Cloud, or if you are self-hosting in single-server mode, the URLs discussed in this page will not be available.

Each PrizmDoc Server instance in a cluster has its own list of the host and port of all servers it considers to be in the cluster (including itself). Each server uses its list when deciding how to route internal traffic within the cluster. In order for your cluster to function correctly, it is important that every server in the cluster always have a correct list of active servers.

There are two cluster management URLs:

- A **GET** which returns the active list of servers a particular server knows about
- A **PUT** which changes the active list of servers a particular server should use

Unlike other PrizmDoc Server APIs, requests to these URLs should be sent directly to individual servers (rather than

to, say, a load balancer which you have put in front of your cluster which would route the request to a random server). Additionally, requests to these URLs must be sent to a server's *public port* (not its *cluster port*, otherwise you will receive `403 Forbidden`).

Whenever a server is added to or removed from your cluster, you should send a `PUT` request to *every* server in the cluster informing them of the new active list of servers they should use.

For more information, see [PrizmDoc Cluster Mode](#).

Available URLs

URL	Description
GET /PCCIS/V1/Service/Properties/Servers	Returns the active list of servers a particular server knows about.
PUT /PCCIS/V1/Service/Properties/Servers	Sets the active list of servers a particular server should use.

GET /PCCIS/V1/Service/Properties/Servers

Gets the active list servers which a particular server is currently using to route requests to. The list of servers returned here must first have been set by a request to `PUT /PCCIS/V1/Service/Properties/Servers` or via the [Central Configuration](#) file.

Successful Response

Response Body

JSON containing the list of host addresses and ports which this server is currently using to route requests to.

- `servers` (Array of Objects) List of servers which this server is currently using to route requests to. Each item in the array will be an object containing:
 - `address` (String) Hostname or IP address where internal traffic should be sent
 - `port` (String) The *cluster port* where internal traffic should be sent (typically "18682"). The cluster port for a server is configured via the `network.clustering.clusterPort` property in the [Central Configuration](#) file.

Error Responses

Status Code	Reason Phrase	Description
403	Forbidden	Can occur if you send the request to a server's <i>cluster port</i> instead of its <i>public port</i> .

Example

In this example:

- The [Central Configuration](#) file uses a `network.publicPort` value of 18681
- The [Central Configuration](#) file uses a `network.clustering.clusterPort` value of 18682 (used by PrizmDoc Services for internal communication between servers)

The `GET` request must be sent to a specific server at its *public port* (18681):

```
GET http://192.168.0.1:18681/PCCIS/V1/Service/Properties/Servers
```

The returned data shows the list of all servers (including the server the request was sent to) with their internal *cluster port* (as a string, "18682"):

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "servers": [
    {
      "address": "192.168.0.1",
      "port": "18682"
    },
    {
      "address": "192.168.0.2",
      "port": "18682"
    },
    {
      "address": "192.168.0.3",
      "port": "18682"
    }
  ]
}
```

PUT /PCCIS/V1/Service/Properties/Servers

Sets the active list of servers which a particular server should start using to route requests to. The list of servers should include the server this request is sent to.

Request

Request Headers

Name	Description
Content-Type	Should be <code>application/json</code>

Request Body

JSON indicating the new active list of servers to use:

- `servers` (Array of Objects) **Required**. List of servers which this server should start using to route requests to. Each item in the array must be an object containing:
 - `address` (String) **Required**. Hostname or IP address where internal traffic should be sent
 - `port` (String) **Required**. The *cluster port* where internal traffic should be sent (typically "18682"). The cluster port for a server is configured via the `network.clustering.clusterPort` property in the [Central Configuration](#) file.

Successful Response

A plain HTTP `200` without any body indicating the new list was accepted and will be used by the server.

Error Responses

Status Code	Reason Phrase	Description
400	Bad Request	The request body could not be understood. Make sure that it is syntactically valid JSON and that it contains the required data.
403	Forbidden	Can occur if you send the request to a server's <i>cluster port</i> instead of its <i>public port</i> .

Example

In this example:

- The [Central Configuration](#) file uses a `network.publicPort` value of 18681
- The [Central Configuration](#) file uses a `network.clustering.clusterPort` value of 18682 (used by PrizmDoc Services for internal communication between servers)

The `PUT` request must be sent to a specific server at its *public port* (18681), but the items in the server list must use the internal *cluster port* (as a string, "18682").

```
PUT http://192.168.0.1:18681/PCCIS/V1/Service/Properties/Servers
Content-Type: application/json

{
  "servers": [
    {
      "address": "192.168.0.1",
      "port": "18682"
    },
    {
      "address": "192.168.0.2",
      "port": "18682"
    },
    {
      "address": "192.168.0.3",
      "port": "18682"
    }
  ]
}
```

```
HTTP/1.1 200 OK
```

The response of HTTP 200 indicates the new list was accepted and will be used by the server.

Viewer Support

These REST APIs are used by PAS and our viewer. It is uncommon for your application to need to use them:

- [Attachments](#)
- [Form Extractors](#)
- [HTML5 Viewing](#)
- [Viewing Sessions](#)

Attachments

Introduction

The *attachments* REST API is used by our viewer to get EML and MSG attachments for a document being viewed.

Available URLs

URL	Description
GET /PCCIS/V1/ViewingSession/u{ViewingSessionID}/Attachments	Lists the attachments available in the source document.

GET /PCCIS/V1/ViewingSession/u{ViewingSessionID}/Attachments

This API returns a JSON list of attachment objects. If the return object list length has a zero count and no errors detected, there are no attachments. The status property of the list object indicates whether or not the list has been completed. If the list hasn't been completed, the request must be retried again. Each Attachment object in the list contains **displayName** and **viewingSessionId** property. The **displayName** can be used to label an href link on a web page and the link points to the **viewingSessionId** URL for the attachment.

Request

URL Parameters

Parameter	Description
{ViewingSessionID}	The ID of the viewing session associated with the request.

Successful Response

Response Body

If successful, this method returns the following properties:

- status** (String) Specifies the current status of the attachments.
 - "pending" - There may be attachments, but the list has not yet been constructed.
 - "complete" - The list is known and present in this object.
- attachments** (List) The list of attachments, if any. Each item in this list will be a new viewing session ID, which is used to view any of the listed attachments in the Viewer by passing in the viewing session ID provided in this list.

Error Responses

Status Code	Description
500	Internal service error. This error can be returned for a number of different reasons. Please verify that your input is correct, and contact Support if the error persists.

Examples

Request

```
GET prizmdoc\_server\_base\_url/PCCIS/V1/ViewingSession/uGcIsIsEGbLV2_V9yy4NzmK2HB-JuLOH--A9sZ16c1a9tx00ZDBGfq1G4kKu0r_GyEps4wWCvDwn4dpnZAR76Uw/Attachments
```

Response

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
```

```
{
  "attachments": [
    {
      "displayName": "example-file.pdf",
      "viewingSessionId":
      "SC0fZEMyIiGdOPMKBqLY8P6EAnVLEqxeTHjUUYqqxgJc3s3wsQ8Lw2qqvkD1uLKTpAlF1ce23EQ6BFpYb4E3LC9TsXxwHCgB-
      I1c5rPOt0"
    },
    {
      "displayName": "second-example.doc",
      "viewingSessionId": "33qQovKTjc0UKbNgOI-5POEyCpNw5x-
      uEzGMB13iUVhnCa_UHSSnOpRBEzPKed7Maxq2RQu2SO0wJj14X4iU_65OQjx2EI5-7h-bX1Yc6uA"
    }
  ],
  "status": "complete"
}
```

Form Extractors

Introduction

The *form extractors* REST API is used by our [e-signature viewer](#) to automatically detect form field elements in a document being viewed.

A *form extractor* resource represents an asynchronous form extraction process. Each *form extractor* that is created is assigned a unique `processId`.

Available URLs

URL	Description
GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/FormInfo	Returns what kind of form field data, if any, is available in a viewing session's source document.
POST /v2/formExtractors	Creates a new <i>form extractor</i> for a work file, starting the process of extracting form field data.
GET /v2/formExtractors/{processId}	Gets the status and final output of a <i>form extractor</i> .

Output Schemas

- `"acroForm"` Output
- `"rasterForm"` Output

GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/FormInfo

Returns what kind of form field data, if any, is available in a viewing session's source document.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.

Successful Response

Response Body

JSON with information about what kind of form data, if any, is available in the source document of the viewing session.

- `formType[]` (Array of strings) Array of values indicating what types of form data, if any, are available for extraction from this viewing session's source document. Values will be one of the following:
 - `"acroForm"` - The source document is a PDF which contains AcroForm data. The data can be extracted by using an `input.formType` of `"acroForm"` in a subsequent POST to create a *form extractor* process.
 - `"xfa"` - The source document is a PDF which contains XFA form data. We do not yet support extraction of XFA data.
 - `"rasterForm"` - The source document is a raster file which may or may not contain detectable form fields. You can attempt to extract form data by using an `input.formType` of `"rasterForm"` in a subsequent POST to create a *form extractor* process.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404		No viewing session with the provided <code>{viewingSessionId}</code> could be found.
480	"DocumentNotProvidedYet"	A source document has not been provided to the viewing session.
480	"FeatureNotLicensed"	You are not licensed to use the form extraction feature.
580	"InternalError"	The server encountered an internal error when handling the request.

Example

Request

```
GET
/PCCIS/V1/ViewingSession/uDLbVh9sTmXJAmd1GeXbS9Gn3WHxs8oib2xPsW2xEFjnIDdoJcudPtxciodSYFQq6zYGabQ_rJIecdbkImTTkSA/FormInfo
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "formType": ["acroform"]
}
```

POST /v2/formExtractors

Creates a new *form extractor* for a work file, starting the process of extracting form field data.

Request

Request Headers

Name	Description
Content-Type	Must be <code>application/json</code>
Accusoft-Affinity-Token	The <code>affinityToken</code> of the work file specified by <code>input.fileId</code> . Required when server clustering is enabled.

Request Body

- `input`
 - `fileId` (String) **Required.** The id of the work file to extract form field data from.
 - `password` (String) Password to open the source document, if required.
 - `formType` (String) **Required.** Type of form field data to extract. Must be one of the following:
 - "acroform" - Extract AcroForm field data from a PDF and return results in our "acroform" JSON format.
 - "rasterForm" - Detect visible form fields in a raster document and return results in our "rasterForm" JSON format.
- `minSecondsAvailable` (Integer) The minimum number of seconds this process will remain available to GET its status. The actual lifetime may be longer. The default lifetime is defined by the `processIds.lifetime` [central configuration](#) parameter.

Successful Response

Response Body

JSON with metadata about the created *form extractor* process. You can check on the status of the form extraction process with additional [GET requests](#).

- `input` (Object) Input we accepted to create the *form extractor* process.
- `processId` (String) Unique id for the newly-created *form extractor* process.
- `affinityToken` (String) Affinity token for this *form extractor*. Present when clustering is enabled.
- `state` (String) State of extracting form field data:
 - "processing" - The server is extracting form field data.
 - "complete" - All form field data has been extracted.
 - "error" - There was a problem extracting form field data.
- `percentComplete` (Integer) Percentage of form extraction which has completed (from 0 to 100).
- `expirationDateTime` (String) Currently planned date and time when the *form extractor* resource will expire and no longer be available. This time may be extended if we have need to keep using the data. Format is [RFC 3339 Internet Date/Time profile of ISO 8601], e.g. "2016-11-05T08:15:30.494Z".
- `errorCode` (String) Descriptive error code. Present when `state` is "error".
- `errorDetails` (Object) Additional error details, if any. May be present when `errorCode` is present.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
400	"MissingInput"	Can occur when clustering is enabled and an <code>Accusoft-Affinity-Token</code> request header was not provided.
480	"MissingInput"	A required input value was not provided. See <code>errorDetails</code> in the response body.
480	"InvalidInput"	An invalid input value was used. See <code>errorDetails</code> in the response body.
480	"FeatureNotLicensed"	You are not licensed to use the form extraction feature.
580	"InternalError"	The server encountered an internal error when handling the request.

Example

Request

```
POST /v2/formExtractors
Content-Type: application/json
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

{
  "input": {
    "fileId": "ek5Zb123oYHSUEVx1bUrVQ",
    "formType": "acroform"
  }
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "fileId": "ek5Zb123oYHSUEVx1bUrVQ",
    "formType": "acroform"
  },
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "state": "processing",
  "percentComplete": 0,
  "expirationDateTime": "2016-12-17T20:38:39.796Z",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

GET /v2/formExtractors/{processId}

Gets the status and final output of a *form extractor*.

Request

URL Parameters

Parameter	Description
{processId}	The <code>processId</code> which identifies the <i>form extractor</i> process.

Request Headers

Name	Description
<code>Accusoft-Affinity-Token</code>	The <code>affinityToken</code> of the form extraction process. Required when server clustering is enabled.

Successful Response

Response Body

JSON with metadata about the *form extractor* process and the final `output`, if available. You can check on the status of the form extraction process with additional GET requests.

- `input` (Object) Input we accepted to create the form extraction process.
- `processId` (String) Unique id for this *form extractor* process.
- `affinityToken` (String) Affinity token for this *form extractor*. Present when clustering is enabled.
- `state` (String) State of extracting form field data:
 - "processing" - The server is extracting form field data.
 - "complete" - All form field data has been extracted.

- "error" - There was a problem extracting form field data.
- percentComplete (Integer) Percentage of form extraction which has completed (from 0 to 100).
- expirationDateTime (String) Currently planned date and time when the *form extractor* resource will expire and no longer be available. This time may be extended if we have need to keep using the data. Format is [RFC 3339 Internet Date/Time profile of ISO 8601], e.g. "2016-11-05T08:15:30.494Z".
- errorCode (String) Descriptive error code. Present when state is "error".
- errorDetails (Object) Additional error details, if any. May be present when errorCode is present.
- output (Object) Present when state is "complete":
 - acroform (Object) Present when input.formType is "acroform". See "acroform" [Output](#) below for details.
 - rasterForm (Object) Present when input.formType is "rasterForm". See "rasterForm" [Output](#) below for details.

Error Responses

Status Code	JSON errorCode	Description
400	"MissingInput"	Can occur when clustering is enabled and an <code>Accusoft-Affinity-Token</code> request header was not provided.
404		No <i>form extractor</i> could be found for the given {processId}.
580	"InternalError"	The server encountered an internal error when handling the request.

Example

Request

```
GET /v2/FormExtractors/gLoltqCVnRKzXz2QFNptqw
Accusoft-Affinity-Token: D+Rmn9kB4FrLfrHoNL2bag6WpuNn2ox2qhT2GbLdf9A=
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "fileId": "-eo_zmq3qmPS0WKZ1P_Lug",
    "formType": "acroform"
  },
  "output": {
    "acroform": {
      "pages": [
        {
          "page": 1,
          "height": 792,
          "width": 612,
          "fields": [
            {
              "fieldType": "Text",
              "name": "email",
              "required": true,
              "readOnly": "true",
              "tabOrder": 0,
              "appearance": {
                "textColor": "0 g",
                "font": "Helvetica"
              },
              "boundingBox": {
                "lowerLeftX": 89,
                "lowerLeftY": 646,
                "upperRightX": 239,
                "upperRightY": 668
              },
              "options": {
                "multiline": false,
                "maxLen": -1
              },
              "format": {
                "formatCategory": "None"
              }
            },
            {
              "fieldType": "Text",
              "name": "fullName",
              "required": false,
              "readOnly": "false",
              "tabOrder": 1,
              "appearance": {
                "textColor": "0 g",
                "font": "Helvetica"
              },
              "boundingBox": {
```

```

        "lowerLeftX": 89,
        "lowerLeftY": 676,
        "upperRightX": 239,
        "upperRightY": 698
      },
      "options": {
        "multiline": false,
        "maxLen": -1
      },
      "format": {
        "formatCategory": "None"
      }
    }
  ]
}
},
"expirationDateTime": "2016-10-11T03:30:33.166Z",
"percentComplete": 100,
"processId": "gLoltqCVnRKzXz2QFNptqw",
"state": "complete",
"affinityToken": "D+Rmn9kB4FrLfrHoNL2bag6WpuNn2ox2qhT2GbLdf9A="
}

```

"acroform" Output

The `output.acroform` object will conform to the following. All properties are always present unless otherwise noted:

- `pages[]` (Array of Objects) Pages in the document which contains acroform fields. Array will be empty if document does not contain any acroform fields. Each item will contain:
 - `page` (Integer) One-indexed page number.
 - `height` (Number) Page height in points.
 - `width` (Number) Page width in points.
 - `fields[]` (Array of Objects) Acroform fields in the current page. Items may contain:
 - `fieldType` (String) Field type. Will be one of the following:
 - "Text" - Text field
 - "Button" - Push button, check box, or radio button:
 - push button when `options.pushButton` is `true`
 - check box when `options.pushButton` and `options.radio` are both `false`
 - radio button when `options.radio` is `true`
 - "Signature" - Signature field
 - `name` (String) Unique field or radio button group name.
 - `required` (Boolean) Indicates whether or not this field is required for the form to be considered complete.
 - `readOnly` (Boolean) Indicates whether or not this field is read only inside the form.
 - `tabOrder` (Integer) Tab order of the field within the document.
 - `boundingBox` (Object) Position and size of this field. Object will contain:
 - `lowerLeftX` (Number) Distance in points from the left edge of the page to the left side of this field.
 - `lowerLeftY` (Number) Distance in points from the bottom edge of the page to the bottom edge of this field.
 - `upperRightX` (Number) Distance in points from the left edge of the page to the right edge of this field.
 - `upperRightY` (Number) Distance in points from the bottom edge of the page to the top edge of this field.
 - `appearance` (Object) Field appearance details:
 - `textColor` (String) Text fill color. Not always present.
 - `font` (String) Font name to use for this field. Not always present.
 - `format` (Object) Field formatting details:
 - `formatCategory` (String) Will be one of the following:
 - "None" - Indicates there are no additional `formatOptions` for this field.
 - "Date" - For text fields, requires the field value to be a date.
 - `formatOptions` Additional options for the given `formatCategory`, if any:
 - When `formatCategory` is "Date": (String) [Date format string](#) to use when formatting the date value for display.
 - `options` (Object) Additional field options, present for some field types:
 - When `fieldType` is "Text":
 - `multiline` (Boolean) Indicates whether or not this is a multi-line text field.
 - `maxLen` (Integer) Indicates the maximum number of characters this form field accepts, or `-1` if there is no limit.
 - When `fieldType` is "Button":
 - `pushButton` (Boolean) `true` if this field is a push button, `false` otherwise.
 - `radio` (Boolean) `true` if this field is a radio button, `false` otherwise.
 - When both `pushButton` and `radio` are `false`, this field is a check box.
 - When `fieldType` is "Button" and `pushButton` is `false`:
 - `buttonOnValue` (String) Indicates the form value to use when this radio button or checkbox is selected/checked.
 - `buttonOffValue` (String) Indicates the form value to use when this radio button or checkbox is not selected/checked. Value will always be "Off".
 - `buttonValue` (String) Indicates whether or not this radio button or checkbox should be initially selected/checked. When the value matches `buttonOnValue`, then this radio button or checkbox should be initially selected/checked. Otherwise (when the value is "Off"), this radio button or checkbox should not be initially selected/checked.

Fill Color Strings

A string of one or more numbers followed by an operator indicating what the numbers represent:

- Grayscale value (when string ends in "g"): A single number between 0 and 1 followed by "g" represents the amount of white which forms a grayscale color value. For example:
 - "0 g" - black
 - "0.5 g" - 50% gray
 - "1 g" - white
- RGB value (when string ends in "rg"): Three numbers between 0 and 1 followed by "rg" represent the amount of red, green, and blue light which are additively mixed to form the final color. For example:
 - "1 0 0 rg" - red
 - "1 1 0 rg" - yellow
 - "0.5 0.25 0.75 rg" - 50% red, 25% blue, 75% green
- CMYK (when string ends in "k"): Four numbers between 0 and 1 followed by "k" represent the amount of cyan, magenta, yellow, and black which should be subtractively mixed to form the final color. For example:
 - "0 0 0 1 k" - black
 - "1 1 1 0 k" - black
 - "1 1 1 1 k" - black
 - "1 0 0 0 k" - cyan
 - "0.25 0.88 0.2 0.16 k" - 25% cyan, 88% magenta, 20% yellow, 16% black

Date Format Strings

Date format strings use the following special substitution patterns:

- `yy` - 2-digit year (e.g. 16 for the year 2016)
- `yyyy` - 4-digit year (e.g. 2016)
- `m` - Month number with no zero padding (e.g. 7 for July)
- `mm` - Month number zero-padded to always be two characters long (e.g. 07 for July)
- `mmm` - Abbreviated month name (e.g. Jan)
- `mmmm` - Full month name (e.g. January)
- `d` - Day of the month with no zero padding (e.g. 4 for the fourth day of the month)
- `dd` - Day of the month zero-padded to always be two characters (e.g. 04 for the fourth day of the month)
- `ddd` - Abbreviated day of the week (e.g. Sun)
- `dddd` - Full name for the day of the week (e.g. Sunday)
- `h` - Hour number in 12-hour time with no zero padding (e.g. 2 for 2 o'clock)
- `hh` - Hour number in 12-hour time zero-padded to always be two characters (e.g. 02 for 2 o'clock)
- `H` - Hour number in 24-hour time with no zero padding (e.g. 13 for the 1:00 pm hour)
- `HH` - Hour number in 24-hour time zero-padded to always be two characters (e.g. 02 for the 2:00 am hour)
- `M` - Minute without zero padding
- `MM` - Minute, zero-padded to always be two digits
- `s` - Second without zero-padding
- `ss` - Second, zero-padded to always be two digits
- `Z` - Offset from UTC (e.g. -0400)
- `j` - Abbreviated Japanese era and year (e.g. H28 for the year 2016).
- `jj` - Full Japanese era and year (e.g. 平成28 for the year 2016).
- `jjj` - Japanese era year without specifying the era (e.g. 28 for the year 2016).

All other characters are considered literal punctuation for the format string. The special characters used above may be used literally by escaping them with a backslash.

"rasterForm" Output

The `output.rasterForm` object will conform to the following. All properties are always present unless otherwise noted:

- `pages[]` (Array of Objects) Information about each page in the raster document. Each item will contain:
 - `page` (Integer) One-indexed page number.
 - `height` (Number) Page height in pixels.
 - `width` (Number) Page width in pixels.
 - `fields[]` (Array of Objects) Fields detected in the current page. Array will be empty if no fields were detected. Items will contain:
 - `name` (String) Unique name we have automatically assigned to this field in the document (e.g. "field5").
 - `fieldType` (String) Field type. Will be one of the following:
 - "Text" - Text field
 - "CheckBox" - Check box
 - `confidence` (Number) Our confidence in the correct detection of this field using a scale of 0 (no confidence) to 100 (complete confidence).
 - `boundingBox` (Object) Position and size of this field. Object will contain:
 - `x` (Number) Distance in pixels from the left edge of the page to the left side of this field.
 - `y` (Number) Distance in pixels from the top edge of the page to the top edge of this field.
 - `width` (Number) Distance in pixels from the left edge of this field (`x`) to the right edge of this field.
 - `height` (Number) Distance in pixels from the top edge of this field (`y`) to the bottom edge of this field.
 - `tables[]` (Array of Objects) Tables detected in the current page. Array will be empty if no tables were detected. Items will contain:
 - `numOfColumns` (Integer) Number of columns in the detected table.
 - `numOfRows` (Integer) Number of rows in the detected table.
 - `fields[]` (Array of Objects) Fields detected in the current table. Items will contain:

- `name` (String) Unique name we have automatically assigned to this field in the document (e.g. "field5").
- `fieldType` (String) Field type. Will be one of the following:
 - "Text" - Text field
 - "CheckBox" - Check box
- `confidence` (Number) Our confidence in the correct detection of this field using a scale of 0 (no confidence) to 100 (complete confidence).
- `boundingBox` (Object) Position and size of this field. Object will contain:
 - `x` (Number) Distance in pixels from the left edge of the page to the left side of this field.
 - `y` (Number) Distance in pixels from the top edge of the page to the top edge of this field.
 - `width` (Number) Distance in pixels from the left edge of this field (`x`) to the right edge of this field.
 - `height` (Number) Distance in pixels from the top edge of this field (`y`) to the bottom edge of this field.

HTML5 Viewing

Introduction

The *HTML5 viewing* REST API is used by PAS and our viewer. It is unusual for your application to need to use this REST API.

Available URLs

URL	Description
GET /PCCIS/V1/Document/q/Attributes	Gets a page count for the source document of a viewing session.
GET /PCCIS/V1/Page/q/{PageNumber}/Attributes	Gets metadata for a page of the source document of a viewing session.
GET /PCCIS/V1/Page/q/{PageNumber}	Gets SVG or an image for a page of the source document of a viewing session.
GET /PCCIS/V1/Page/q/{PageNumber}/Tile/{x}/{y}/{width}/{height}	Gets a "tile" image, a part of a page, for a page of the source document of a viewing session.
GET /PCCIS/V1/Page/q/{PageNumber}/Thumbnail	Gets a thumbnail image for a page of the source document of a viewing session.
GET /PCCIS/V1/Document/q/{PageNumberBegin}-{PageNumberEnd}/Text	Gets currently-available text and text metadata for a range of pages for the source document of a viewing session.
GET /v2/viewingSessions/{viewingSessionId}/revisionData	Gets objects which describe known changes between the two documents used as input to a comparison viewing session.

Deprecated URLs

URL	Description
GET /PCCIS/V1/License/ClientViewer	Deprecated.

GET /PCCIS/V1/Document/q/Attributes?DocumentID=[e,u]{ViewingSessionId}&DesiredPageCountConfidence={DesiredPageCountConfidence}

Gets a page count for the source document of a viewing session.

Request

Query String Parameters

Parameter	Description
<code>DocumentID</code>	Required. The <code>viewingSessionId</code> which identifies the viewing session, prefixed with <code>u</code> if in <i>unencoded</i> plaintext form or prefixed with <code>e</code> if <i>base-64 encoded</i> .
<code>DesiredPageCountConfidence</code>	An integer from 0 and 100 inclusive which specifies the minimum required confidence in the page count before a value is returned. A value of 50 or lower is more likely to result in an estimated page count being returned. Default is 100, requiring that the actual page count be returned.

Successful Response

JSON metadata about the document page count.

- `pageCount` (Integer) - Currently determined number of pages in the document.
- `pageCountConfidence` (Integer) - An integer from 0 to 100 indicating the confidence that `pageCount` is accurate. When less than 100, `pageCount` is still being determined and the current value should be considered an estimate. You can repeat the request to see if a more accurate `pageCount` is available. When 100, the `pageCount` has been finalized and the value should be considered the actual number of pages in the document.

Example

```
GET prizmdoc_server_base_url/PCCIS/V1/Document/q/Attributes?DocumentID=uXYZ...
```

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8
```

```
{
  "pageCount": 3,
  "pageCountConfidence": 100
}
```

GET /PCCIS/V1/Page/q/{PageNumber}/Attributes?DocumentID=[e,u]{viewingSessionId}&ContentType={ContentType}

Gets metadata for a page of the source document of a viewing session.

Request

URL Parameters

Parameter	Description
{PageNumber}	Zero-indexed page number to get information about.

Query String Parameters

Parameter	Description
DocumentID	Required. The <code>viewingSessionId</code> which identifies the viewing session, prefixed with <code>u</code> if in <i>unencoded</i> plaintext form or prefixed with <code>e</code> if base-64 <i>encoded</i> .
ContentType	Used to indicate whether you want attributes for SVG page content or raster page content. Use <code>svg</code> (or <code>svga</code> or <code>svgb</code>) to get page attributes for SVG content or <code>png</code> to get page attributes for raster content. Default is <code>svg</code> .

Successful Response

JSON metadata about the page.

- `version` (String) - *Deprecated*. Value will always be "7.1".
- `contentType` (String) - Comma-separated types of content available, hard-coded to a specific set of values depending on the requested `ContentType`:
 - Will be "jpeg,png,svg" when requested `ContentType` is `svg` (or `svga`, or `svgb`)
 - Will be "jpeg,png" when requested `ContentType` is `png`
- `imageBitDepth` (Integer) - Bit depth of raster content. Only relevant when requested `ContentType` is `png`. When requested `ContentType` is `svg` (or `svga` or `svgb`), the value will be hard-coded to 16.
- `imageHeight` (Integer) - Height of the page:
 - in *pixels* when requested `ContentType` is `png`
 - in *unspecified units* when requested `ContentType` is `svg` (or `svga` or `svgb`)
- `imageWidth` (Integer) - Width of the page:
 - in *pixels* when requested `ContentType` is `png`
 - in *unspecified units* when requested `ContentType` is `svg` (or `svga` or `svgb`)
- `imageXResolution` (Integer) - Relative horizontal resolution of raster content when requested `ContentType` is `png` (like pixels per inch except that the unit is unspecified and will not necessarily be inches). When requested `ContentType` is `svg` (or `svga` or `svgb`), the value will be hard-coded to 90.
- `imageYResolution` (Integer) - Relative vertical resolution of raster content when requested `ContentType` is `png` (like pixels per inch except that the unit is unspecified and will not necessarily be inches). When requested `ContentType` is `svg` (or `svga` or `svgb`), the value will be hard-coded to 90.

Examples

Get attributes for the SVG form page 0. The two most-valuable properties are "`imageHeight`" and "`imageWidth`" which indicate the width and height of the SVG page in unspecified units. The rest are hard-coded values:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8

{
  "version": "7.1",
  "contentType": "jpeg,png,svg",
  "imageBitDepth": 16,
  "imageHeight": 842,
  "imageWidth": 595,
  "imageXResolution": 90,
  "imageYResolution": 90
}
```

Get attributes for the raster form of page 0:

```
GET prizmdoc_server_base_url/PCCIS/V1/Page/q/0/Attributes?DocumentID=uXYZ...?ContentType=png
```

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8

{
  "version": "7.1",
  "contentType": "jpeg,png",
  "imageBitDepth": 8,
  "imageHeight": 1755,
  "imageWidth": 1240,
  "imageXResolution": 150,
  "imageYResolution": 150
}
```

GET /PCCIS/V1/Page/q/{PageNumber}?DocumentID=[e,u]{ViewingSessionId}&Scale={Scale}&ContentType={ContentType}

Gets SVG or an image for a page of the source document of a viewing session.

Request

Request Headers

Name	Description
Accept-Encoding	Specify <code>gzip</code> to allow gzip compression of the response. Gzip compression will only be applied to SVG responses (it is not used for PNG and JPEG responses) and it may be skipped if the SVG is small. If a response is compressed it will contain a <code>Content-Encoding: gzip</code> response header. Because all modern browsers support <code>Content-Encoding: gzip</code> responses, we recommend you always provide an <code>Accept-Encoding: gzip</code> request header.

URL Parameters

Parameter	Description
{PageNumber}	Zero-indexed page number whose content should be returned.

Query String Parameters

Parameter	Description
DocumentID	Required. The <code>viewingSessionId</code> which identifies the viewing session, prefixed with <code>u</code> if in <i>unencoded</i> plaintext form or prefixed with <code>e</code> if <i>base-64 encoded</i> .
ContentType	Type of content to be returned. Default is <code>png</code> . Possible values: <ul style="list-style-type: none"> <code>svgb</code> - Fully-optimized SVG (uses a unicode inline font to store glyph definitions). Smallest possible SVG, but may not be compatible with some browsers. Recommended whenever possible. <code>svga</code> - Partially-optimized SVG (uses a non-unicode inline font to store only the most frequently-occurring glyph definitions). May not be compatible with some browsers. Use only if <code>svgb</code> content is not compatible with the target browser. <p>NOTE: <i>PrizmDoc ViewerControl falls back to svga Content-Type if svgb content is not compatible with the browser.</i></p> <code>svg</code> - Unoptimized SVG (no font is used; glyph definitions are expressed as SVG path operations). Broadest compatibility with browsers but typically <i>much</i> larger, so it renders and scrolls much slower than <code>svgb</code> and <code>svga</code>. Not recommended. Use only as a fallback if both <code>svgb</code> and <code>svga</code> are not compatible with the target browser, or the use of webfonts is disabled in the target browser. <p>NOTE: <i>PrizmDoc ViewerControl falls back to svg Content-Type if both svgb and svga are not compatible with the browser or the use of webfonts is disabled.</i></p> <code>png</code> - PNG image. <code>jpeg</code> - JPEG image.
Scale	Scaling factor to apply when returning a PNG or JPEG image. The image will be resized by multiplying its width and height by this value. A value of <code>1.0</code> leaves the image unscaled, values less than <code>1.0</code> make the image smaller, and values greater than <code>1.0</code> make the image larger. For example, a value of <code>2.0</code> would return an image whose width and height have been doubled. A value of <code>0.5</code> would return an image whose width and height have been halved. Only applies when <code>ContentType</code> is <code>png</code> or <code>jpeg</code> , ignored otherwise. Default is <code>1.0</code> (no scaling applied).

Successful Response with Page Content

Response Headers

Name	Description
Content-Type	The type of content returned. Possible values: <ul style="list-style-type: none"> <code>image/svg+xml</code> - When the request query string parameter <code>ContentType</code> was <code>svg</code>, <code>svga</code>, or <code>svgb</code>. <code>image/png</code> - When the request query string parameter <code>ContentType</code> was <code>png</code>. <code>image/jpeg</code> - When the request query string parameter <code>ContentType</code> was <code>jpeg</code>.
Content-Encoding	May be set to <code>gzip</code> if the request used <code>Accept-Encoding: gzip</code> .
Accusoft-Data-Encrypted	Indicates whether or not page content has been encrypted. <code>true</code> when page content is encrypted, <code>false</code> otherwise. See Enabling Content Encryption .

Response Body

SVG, PNG, or JPEG for the requested page.

Examples

Get page 0 as SVG

Request:

```
GET prizmdoc_server_base_url/PCCIS/V1/Page/q/0?DocumentID=uXYZ...&ContentType=svgb
```

Response:

```
HTTP/1.1 200 OK
Content-Type: image/svg+xml
Content-Encoding: gzip
Accusoft-Data-Encrypted: false

<svg height="842" style="font-family:qsnvcgduoqekywbefqyyjjhodpw;font-size:12px;" version="1.2" viewBox="0 0 595 842" width="595"
  xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
  ...
</svg>
```

Get page 0 as a PNG

```
GET prizmdoc_server_base_url/PCCIS/V1/Page/q/0?DocumentID=uXYZ...&ContentType=png
```

Response:

```
HTTP/1.1 200 OK
Content-Type: image/png
Accusoft-Data-Encrypted: false

<<PNG bytes>>
```

Response When SVG Content is Not Available

SVG content is typically preferred but not always available. For example, if the source document is raster (such as a TIFF), then only raster page content (PNG and JPEG) will be available. It is common for a client viewer to always try and request SVG content first. Then, once it becomes clear SVG is not available, the client viewer can fallback to only requesting PNG or JPEG page content.

For this reason, if SVG is requested but is not available, we respond with a successful HTTP 200 but with a JSON body indicating that SVG is not available. Additionally, we include raster page attributes metadata in the JSON so that the viewer does not need to issue an additional [request for page attributes](#).

Example

Request:

```
GET prizmdoc\_server\_base\_url/PCCIS/V1/Page/q/0?DocumentID=uXYZ...&ContentType=svgb
```

Response when SVG is not available:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8

{
  "errorCode": "SvgNotAvailable",
  "pageAttributes": {
    "version": "7.1",
    "contentType": "jpeg,png",
    "imageBitDepth": 8,
    "imageHeight": 280,
    "imageWidth": 593,
    "imageXResolution": 72,
    "imageYResolution": 72
  }
}
```

GET /PCCIS/V1/Page/q/{PageNumber}/Tile/{X}/{Y}/{Width}/{Height}?DocumentID=[e,u]{ViewingSessionId}&ContentType={ContentType}&Scale={Scale}

Gets a "tile" image, a part of a page, for a page of the source document of a viewing session.

Request

URL Parameters

Parameter	Description
{PageNumber}	Zero-indexed page number to extract a tile from.
{X}	Where the left edge of the tile should begin, expressed as the number of pixels from the left edge of the page. Must be less than the pixel width of the (scaled) page.
{Y}	Where the top of the tile should begin, expressed as the number of pixels from the top edge of the page. Must be less than the pixel height of the (scaled) page.
{Width}	Width of the tile in pixels, extended right from {X}. If the right edge of the tile extends beyond the right edge of the page then the tile will be trimmed so that only actual page content is returned. You can safely use a {Width} value that goes beyond the right edge of the page, but note that the actual width of the returned image may be different than what you requested.
{Height}	Height of the tile in pixels, extended down from {Y}. If the bottom edge of the tile extends beyond the bottom edge of the page then the tile will be trimmed so that only actual page content is returned. You can safely use a {Height} value that goes beyond the bottom edge of the page, but note that the actual height of the returned image may be different than what you requested.

Query String Parameters

Parameter	Description
DocumentID	Required. The <code>viewingSessionId</code> which identifies the viewing session, prefixed with <code>u</code> if in <i>unencoded</i> plaintext form or prefixed with <code>e</code> if <i>base-64 encoded</i> .
ContentType	Type of image to be returned. Default is <code>png</code> . Possible values: <ul style="list-style-type: none"><code>png</code><code>jpeg</code> - Not recommended if you are requesting multiple tiles to be "stitched" together due to alignment artifacts that will occur at tile boundaries.
Scale	Scaling factor to apply to the entire page <i>before</i> cropping to the specified tile region. The full page image will be resized by multiplying its width and height by this value. A value of <code>1.0</code> leaves the page unscaled, values less than <code>1.0</code> make the page smaller, and values greater than <code>1.0</code> make the page larger. Default is <code>1.0</code> (no scaling applied).

Successful Response

Response Headers

Name	Description
Content-Type	The type of content returned. Possible values: <ul style="list-style-type: none"> <code>image/png</code> - When the request query string parameter <code>ContentType</code> was <code>png</code>. <code>image/jpeg</code> - When the request query string parameter <code>ContentType</code> was <code>jpeg</code>.
Accusoft-Data-Encrypted	Indicates whether or not page content has been encrypted. <code>true</code> when page content is encrypted, <code>false</code> otherwise. See Enabling Content Encryption .

Response Body

Tile image.

Example

Request a 512x512 PNG tile for page 0 starting at x-position 1024 (from left) and y-position 1536 (from top):

```
GET prizmdoc_server_base_url/PCCIS/V1/Page/q/0/Tile/1024/1536/512/512?DocumentID=uXYZ...
```

```
HTTP/1.1 200 OK
Content-Type: image/png

<<PNG bytes>>
```

GET /PCCIS/V1/Page/q/{PageNumber}/{Width}x{Height}?DocumentID=[e,u]{ViewingSessionId}&ContentType={ContentType}

Gets a thumbnail image for a page of the source document of a viewing session.

The page will be resized, maintaining aspect ratio, to fit within the `{Width}` and `{Height}` specified in the URL.

Request**URL Parameters**

Parameter	Description
<code>{PageNumber}</code>	Zero-indexed page number to create a thumbnail image for.
<code>{Width}</code>	Maximum allowed width, in pixels, of the thumbnail. Must be an integer greater than 0.
<code>{Height}</code>	Maximum allowed height, in pixels, of the thumbnail. Must be an integer greater than 0.

Query String Parameters

Parameter	Description
<code>DocumentID</code>	Required. The <code>viewingSessionId</code> which identifies the viewing session, prefixed with <code>u</code> if in <i>unencoded</i> plaintext form or prefixed with <code>e</code> if <i>base-64 encoded</i> .
<code>ContentType</code>	Type of image to be returned. Default is <code>png</code> . Possible values: <ul style="list-style-type: none"> <code>png</code> <code>jpeg</code>

Successful Response**Response Headers**

Name	Description
Content-Type	The type of content returned. Possible values: <ul style="list-style-type: none"> <code>image/png</code> - When the request query string parameter <code>ContentType</code> was <code>png</code>. <code>image/jpeg</code> - When the request query string parameter <code>ContentType</code> was <code>jpeg</code>.

Response Body

Thumbnail image.

Example

Request a thumbnail image of page 0 that fits within a 200x200 square:

```
GET prizmdoc_server_base_url/PCCIS/V1/Page/q/0/200x200?DocumentID=uXYZ...
```

```
HTTP/1.1 200 OK
Content-Type: image/png

<<PNG bytes>>
```

GET /PCCIS/V1/Document/q/{PageNumberBegin}-{PageNumberEnd}/Text?DocumentID=[e,u]{ViewingSessionId}

Gets currently-available text and text metadata for a range of pages for the source document of a viewing session.

NOTE: This URL is designed to support our viewer. If you want to simply programmatically extract text from a document, use the [Search Contexts API](#) instead, specifically `POST /v2/searchContexts` and `GET /v2/searchContexts/{contextId}/records`.

Request

URL Parameters

Parameter	Description
{PageNumberBegin}	Zero-indexed page number of the first page in the range.
{PageNumberEnd}	Zero-indexed page number of the last page in the range.

Query String Parameters

Parameter	Description
DocumentID	Required. The <code>viewingSessionId</code> which identifies the viewing session, prefixed with <code>u</code> if in <i>unencoded</i> plaintext form or prefixed with <code>e</code> if <i>base-64 encoded</i> .

Request Headers

Name	Description
Accept-Encoding	Specify <code>gzip</code> to allow gzip compression of the response. Gzip compression may be skipped if the overall response size is small. If a response is compressed it will contain a <code>Content-Encoding: gzip</code> response header. Because all modern browsers support <code>Content-Encoding: gzip</code> responses, we recommend you always provide an <code>Accept-Encoding: gzip</code> request header.

Successful Response

Response Headers

Name	Description
Content-Encoding	May be set to <code>gzip</code> if the request used <code>Accept-Encoding: gzip</code> .
Accusoft-Data-Encrypted	Indicates whether or not page content has been encrypted. <code>true</code> when page content is encrypted, <code>false</code> otherwise. See Enabling Content Encryption .

Response Body

JSON containing page text and text positioning metadata.

- pages[]** (Array of Objects) **Always present in case of success. Optional in case of failure.** Will contain an array of objects, each containing text data for a page, for pages where text has been successfully extracted. Note, however, that text extraction takes time and text may not yet be available for the range of pages requested. If the array is empty or contains fewer items than the number of pages included in your page range, then the text for the requested page range has not been fully extracted. Repeating the request should eventually produce an array with the expected number of items. Note also that the order of the records is not guaranteed; you must use the `number` property of each returned item to know its page index. Items may contain:
 - `number` (Integer) **Always present.** Page index (zero-indexed page number). The property is named simply `number` for backwards compatibility reasons.
 - `text` (String) Page text.
 - `errorCode` (Integer) When text cannot be extracted for this page, present with a value of `1`.
 - `errorDescription` (String) When text cannot be extracted for this page, a descriptive error message explaining why (such as "No page data was found.") or an empty string if the cause of the error is unknown.
 - `width` (Number) Page width.
 - `height` (Number) Page height.
 - `rectangles[]` (Array of Arrays) Bounding boxes for individual glyphs on the page. Each item will contain four numbers:
 - `[0]` (Number) Distance from the left edge of the page to the left edge of the glyph bounding box.
 - `[1]` (Number) Distance from the top edge of the page to the top edge of the glyph bounding box.
 - `[2]` (Number) Width of the glyph bounding box.
 - `[3]` (Number) Height of the glyph bounding box.
 - `markup[]` (Array of Objects) Objects describing hyperlinks, if any. Each item may contain:
 - `changeType` (String) Value will always be "Add".
 - `markType` (String) Value will always be "DocumentHyperlink".
 - `properties` (Object) Properties of the hyperlink.
 - `href` (String) Destination URL.
 - `rectangle` (Object) Dimensions of the hyperlink bounding box on the page.
 - `x` (Number) Distance from the left edge of the page to the left edge of the hyperlink bounding box.
 - `y` (Number) Distance from the top edge of the page to the top edge of the hyperlink bounding box.
 - `width` (Number) Width of the hyperlink bounding box.
 - `height` (Number) Height of the hyperlink bounding box.
 - `borderThickness` (Number) Border thickness which should be applied.
 - `borderHorizontalRadius` (Number) Horizontal border radius which should be applied.
 - `borderVerticalRadius` (Number) Vertical border radius which should be applied.
 - `borderOpacity` (Integer) Border opacity which should be applied. Value will be from `0` to `255`, where `0` represents fully transparent and `255` represents fully opaque.
 - `errorCode` (Integer) **Might be present in case of failure and missing pages[] object.** When text cannot be extracted for this page, present with a value of `1`.
 - `errorDescription` (String) **Might be present in case of failure and missing pages[] object.** When text cannot be extracted for this page, a descriptive error message explaining why (such as "No page data was found.") or an empty string if the cause of the error is unknown.

Examples

Request text for pages 0 through 9:

```
GET /prizmdoc_server_base_url/PCCIS/V1/Document/q/0-9/Text?DocumentID=xXYZ...
```

Response when text is not yet available:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8
```

```
Accusoft-Data-Encrypted: false
```

```
{
  "pages": []
}
```

Response when text is available (where ... indicates that data has been omitted for brevity):

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "pages": [
    {
      "number": 0,
      "text": "the page text",
      "width": 648.00,
      "height": 828.00,
      "rectangles": [
        [
          202.25,
          135.05,
          27.00,
          73.26
        ],
        [
          229.25,
          135.05,
          30.00,
          73.26
        ],
        ...
      ]
    },
    {
      "markup": [
        {
          "changeType": "Add",
          "markType": "DocumentHyperlink",
          "properties": {
            "rectangle": {
              "height": 14.71,
              "width": 86.20,
              "y": 73.50,
              "x": 71.31
            },
            "borderHorizontalRadius": 0.0,
            "borderVerticalRadius": 0.0,
            "borderThickness": 0.0,
            "href": "http://www.google.com/",
            "borderOpacity": 255
          }
        },
        ...
      ]
    },
    ...
  ]
}
```

GET /v2/viewingSessions/{viewingSessionId}/revisionData?limit={limit}&continueToken={continueToken}

Gets objects which describe known changes between the two documents used as input to a comparison viewing session.

This URL is designed to give you an array of changes in chunks as the individual changes become available. Each GET request will return the currently-known changes up to a limit (default is 100). If a response contains a `continueToken`, it indicates that additional changes may be available and that you should issue another GET request using that `continueToken` as a query string parameter to skip the changes you have already received. As long as a response contains a `continueToken`, use it to issue a subsequent GET for more changes. When you encounter a response which does not have a `continueToken`, you have received all of the changes and no more GET requests are necessary.

In order to optimize the number of network requests you make, any response which contains a `continueToken` will also contain a `continueAfter` value with a recommended number of milliseconds you should wait before sending the next GET request.

Request

URL Parameters

Parameter	Description
{limit}	The maximum number of changes to return for this HTTP request. Must be an integer greater than 0. Default is 100.
{continueToken}	Used to continue getting changes from the point where a previous GET request left off.

Request Headers

Name	Description
Accept-Encoding	Set to <code>gzip</code> to get a gzipped response body.

Successful Response

Response Headers

Name	Description
Content-Encoding	Will be set to <code>gzip</code> if the request used <code>Accept-Encoding: gzip</code>

Response Body

JSON with any available `changes`.

- `changes` (Array of Objects) **Always present**. Array of newly-available changes, objects which each describe a difference between the two documents being compared. If no new changes are available, this array will be empty.
 - `id` (Integer) Unique number assigned to this change.
 - `endPageIndex` (Integer) Zero-indexed page number where this change ends in the document.
 - `type` (String) Type of the change. Will be one of the following:
 - "contentInserted"
 - "contentDeleted"
 - "propertyChanged"
 - "paragraphNumberChanged"
 - "paragraphPropertyChanged"
 - "tablePropertyChanged"
 - "sectionPropertyChanged"
 - "styleDefinitionChanged"
 - "contentMovedFrom"
 - "contentMovedTo"
 - "tableCellInserted"
 - "tableCellDeleted"
 - "tableCellsMerged"
- `continueToken` (String) When present, indicates that more changes may be available. An additional GET request should be made for more changes using this value as the `continueToken` query string parameter. When not present, indicates that all changes have been obtained and no further changes will be available.
- `continueAfter` (Number) Recommended milliseconds to delay before issuing the next GET request for more changes.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No viewing session with the provided <code>viewingSessionId</code> could be found.
480	"InvalidInput"	An invalid input value was used. See <code>errorDetails</code> in the response body.
580	"InternalError"	The server encountered an internal error when handling the request.

Example

Here is an example sequence of requests and responses illustrating how you would acquire the full set of changes for a comparison viewing session (for brevity, the total number of changes in this example is small).

You would start with an initial GET:

```
GET
prizmdoc_server_base_url/v2/viewingSessions/luMJZGieGQr20veY15JQwsv77iIvaFsvHAW4x1L881kBs8mk63aArufxZ9jaXZ0ykG5LsMlWorI6u3Ui6YApkw/revisionData
Accept-Encoding: gzip
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Encoding: gzip

{
  "changes": [],
  "continueToken": "luMJZGieGQr20veY15JQwsv77iIvaFsvHAW4x1L881mZwRo30ojTLjaT0J2D2f8D",
  "continueAfter": 500
}
```

In this case, the initial response did not return any changes at all (the `changes` array is empty), but the presence of a `continueToken` indicates they may simply not have been available yet. We should issue another GET request after waiting 500 milliseconds (the amount of time recommended by `continueAfter`).

So, half a second later, we issue a follow-up request with the `continueToken` passed in as a query string parameter:

```
GET
prizmdoc_server_base_url/v2/viewingSessions/luMJZGieGQr20veY15JQwsv77iIvaFsvHAW4x1L881kBs8mk63aArufxZ9jaXZ0ykG5LsMlWorI6u3Ui6YApkw/revisionData?
continueToken=luMJZGieGQr20veY15JQwsv77iIvaFsvHAW4x1L881mZwRo30ojTLjaT0J2D2f8D
Accept-Encoding: gzip
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Encoding: gzip

{
  "changes": [
    {
      "id": 0,
      "endPageIndex": 0,
      "type": "contentInserted"
    },
    {
      "id": 1,
      "endPageIndex": 0,
      "type": "contentDeleted"
    }
  ],
  "continueToken": "luMJZGieGQr20veY15JQwsv77iIvaFsvHAW4x1L881klhqP2L79Yero0nM9aoZ9r",
  "continueAfter": 500
}
```

This time we receive two changes. The presence of a new `continueToken` tells us there may be more, so we submit another request with the new `continueToken`.

Notice in the next response that the changes which have already been given to us are not repeated:

```
GET
prizmdoc_server_base_url/v2/viewingSessions/luMJZGieGQr20veY15JQwsv77iIvaFsvHAW4x1L881kBs8mk63aArufxZ9jaX20ykG5LsMlWorI6u3Ui6YApkw/revisionData?
continueToken=luMJZGieGQr20veY15JQwsv77iIvaFsvHAW4x1L881k1hqP2L79Yero0nM9aoZ9r
Accept-Encoding: gzip
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Encoding: gzip
```

```
{
  "changes": [
    {
      "id": 2,
      "endPageIndex": 5,
      "type": "styleChanged"
    }
  ]
}
```

This time we get a new change, and the lack of a `continueToken` tells us we have received all of the changes, so there are no more GET requests to make.

GET /PCCIS/V1/License/ClientViewer

NOTE: This URL has been deprecated and will be removed from the public API in a future release. It no longer functions and returns HTTP 500 *Internal Server Error*.

Viewing Sessions

Introduction

The PrizmDoc Server *viewing sessions* REST API is used by PAS and is maintained for backwards compatibility. It should not be used for new application development. **Please use the newer [PAS viewing sessions REST API](#) instead.**

Available URLs

URL	Description
POST /PCCIS/V1/ViewingSession	Creates a new viewing session.
GET /PCCIS/V1/ViewingSession/u{viewingSessionId}	Gets information about a viewing session.
PUT /PCCIS/V1/ViewingSession/u{viewingSessionId}/SourceFile	Uploads the source document to be used for a viewing session.
PUT /v2/viewingSessions/{viewingSessionId}/sourceFile/original	When viewing a comparison of two source documents, uploads the first of the two source documents.
PUT /v2/viewingSessions/{viewingSessionId}/sourceFile/revised	When viewing a comparison of two source documents, uploads the second of the two source documents.
PUT /PCCIS/V1/ViewingSession/u{viewingSessionId}/SourceRef	Attach an existing work file (or, when doing comparison, two work files) as the source document to be used for a viewing session.
GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/SourceFile	Downloads the source document in use for a viewing session.
GET /v2/viewingSessions/{viewingSessionId}/sourceFile/original	When viewing a comparison of two documents, downloads the first of the two source documents.
GET /v2/viewingSessions/{viewingSessionId}/sourceFile/revised	When viewing a comparison of two documents, downloads the second of the two source documents.
GET /v2/viewingSessions/{viewingSessionId}/restrictions	Returns information about any restrictions enforced by the server for the current viewing session.

URL	Description
GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/FileId	Gets the work file id for the source document in use for a viewing session.
POST /PCCIS/V1/ViewingSession/u{viewingSessionId}/Notification/SessionStarted	Ensures PrizmDoc Server has started the process of converting a viewing session's source document to HTML.
POST /PCCIS/V1/ViewingSession/u{viewingSessionId}/Notification/SessionStopped	Invalidates a viewing session so that it can no longer be used.
POST /PCCIS/V1/ViewingSession/u{viewingSessionId}/Notification/SessionErrored	Marks a viewing session as errored so that it can no longer be used.
POST /PCCIS/V1/ViewingSession/u{viewingSessionId}/Replacement	Replaces a viewing session with new parameters.
DELETE /PCCIS/V1/ViewingSession/u{viewingSessionId}	Deletes a viewing session.

POST /PCCIS/V1/ViewingSession

Creates a new viewing session. At a high level, a viewing session takes a source document as input and produces HTML page content and document text as output.

Request

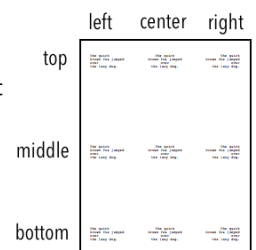
Request Headers

Name	Description
Content-Type	Should be <code>application/json</code>
Accusoft-Affinity-Hint	Some sort of string value which uniquely identifies the source document you intend to use for the viewing session. Used to ensure that two viewing sessions created with the same <code>Accusoft-Affinity-Hint</code> are very likely to be handled by the same server, increasing the chances that cached output is reused for viewing sessions of the same source document. See Optimizing Cache Performance for Cluster Mode .
Accusoft-Affinity-Token	When you intend to use an existing work file as the source document via a subsequent call to <code>PUT /SourceRef</code> , the <code>affinityToken</code> of the existing work file. Required only when you intend to use an existing work file as the source document.

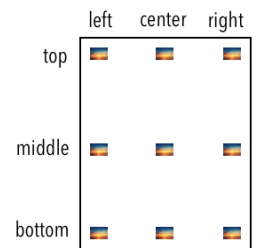
Request Body

- `minSecondsAvailable` (Integer) Minimum number of seconds the viewing session will exist or `0` to use the default value specified by product configuration. This property is only really useful if you explicitly set `serverCaching` to `"none"`. When `serverCaching` is `"full"`, the only allowed value for this property is `0`. Default is `0`.
- `documentSource` (String) Specifies how the source document will be provided. Default is `"api"`. Possible values:
 - `"api"` - The source document will be provided by the calling application in a subsequent API request.
 - `"http"` - PrizmDoc Server will download the source document from a URL specified by `externalId`.
 - `"file"` - PrizmDoc Server will use a local file as the source document using the file path specified by `externalId`. > **NOTE: By default, `"file"` is not enabled as a valid `documentSource`. Enable `"file"` by adding it to the `viewing.sessionConstraints.documentSource.allowedValues` array in [Central Configuration](#).**
- `externalId` (String) **Required when `documentSource` is `"http"` or `"file"`.** Indicates where the source document should be acquired from:
 - *When `documentSource` is `"http"`:* A URL where the source document will be downloaded from.
 - *When `documentSource` is `"file"`:* A path to a file on the server which will be used as the source document.
- `documentExtension` (String) - File extension of the source document (like `"docx"`, `"html"`, or `"csv"`) used to indicate the source document file format. Often unnecessary. Only required when 1) the `documentSource` is `"http"` or `"file"` but the `externalId` containing the URL or file path did not end with a recognizable file extension and 2) the source document file format could not be automatically detected (this most-commonly occurs for text-based file formats, such as `txt`, `csv`, and

- html). > **NOTE:** If Format Detection is disabled, then the source document format will never be automatically detected.
- `password` (String) Password to use when opening a password-protected source document.
 - `tenantId` (String) Custom, arbitrary tenant id to be associated with the viewing session. PrizmDoc Server has no concept of tenants; if provided, this metadata is only for use by the calling application.
 - `origin` (Object) Custom, arbitrary set of key/value string pairs to be associated with the viewing session. Intended for associating end user origin data (like IP address or hostname), but you can use any set of key/value strings you want.
 - `render` (Object) Options which control how browser content is rendered:
 - `html5` (Object) Options when the output format is HTML:
 - `alwaysUseRaster` (Boolean) **Required.** Determines whether only raster data, instead of SVG, should be created for the viewing session. With modern browsers, it is rare to only want raster. This is typically set to `false`.
 - `svgMaxImageSize` (Number) The maximum edge length, in pixels, that is allowable for any image when creating SVG. For example, a value of `8000` would ensure that any images in a PDF whose width or height were greater than 8000 pixels would be down-sampled before the image was added to the final SVG. Default is configurable, but is typically `8000`. To disable this optimization, use a value of `0`.
 - `vectorTolerance` (Number) For CAD documents, the amount of path simplification that is allowable when creating the SVG. Path simplification will merge points which are "close together" to create optimized SVG. You can think of this value as defining what "close together" means. Higher values introduce more simplification but also more distortion. Default is configurable, but is typically `0.3`. Cannot be greater than `10.0`. To disable this optimization, use a value of `0`.
 - `rasterResolution` (Integer) *Deprecated.* Providing this value no longer has any effect.
 - `flash` (Object) *Deprecated.* We no longer produce Flash content. Providing these options no longer has any effect.
 - `optimizationLevel` (Integer) *Deprecated.* Providing this value no longer has any effect.
 - `watermarks` (Array of Objects) Objects describing watermarks which should be applied to page content. Each item must be an object which conforms to the following:
 - `text watermark`:
 - `type`: `"text"` (String) **Required.** Must be set to `"text"` to indicate the object represents a text watermark.
 - `text` (String) Actual text of the watermark. Within the string, you can use the following special tokens to insert dynamic values:
 - `{{pageNumber}}` - Will be replaced with the current page number.
 - `{{pageCount}}` - Will be replaced with the total number of pages.
 - `opacity` (Number) Opacity of the watermark. `1.0` is completely opaque, `0.0` is completely transparent. Default is `1.0`.
 - `color` (String) Text color. Can be any valid CSS color name (like `"red"`) or hex value (like `"#FF0000"`). Default is `"black"`.
 - `fontFamily` (String) Font family for the text. Default for SVG output is to use the browser's default font. Default for raster output is unspecified.
 - `fontSize` (String) Font size specified in points (like `"12pt"`). Default for SVG output is to use the browser's default font size. Default for raster output is unspecified.
 - `fontWeight` (String) Determines the font weight. Possible values:
 - `"normal"` (default)
 - `"bold"`
 - `fontStyle` (String) Determines the font style. Possible values:
 - `"normal"` (default)
 - `"italic"`
 - `textDecoration` (String) Possible values:
 - `"none"` (default)
 - `"underlined"`
 - `horizontalAlign` (String) Determines the horizontal position of the watermark. Default is `"center"`. Possible values:
 - `"left"` - Text will be horizontally anchored to the left side of the page and text will be left aligned.
 - `"center"` - Text will be horizontally anchored to the center of the page and text will be centered. (default)
 - `"right"` - Text will be horizontally anchored to the right side of the page and text will be right aligned.
 - `verticalAlign` (String) Determines the vertical position of the watermark. Default is `"middle"`. Possible values:
 - `"top"` - Text will be vertically anchored to the top of the page.



- "middle" Text will be vertically anchored to the middle of the page. (default)
 - "bottom" - Text will be vertically anchored to the bottom of the page.
 - *diagonal text watermark:*
 - type: "diagonalText" (String) **Required.** Must be set to "diagonalText" to indicate the object represents a diagonal text watermark.
 - text (String) Actual text of the watermark. Within the string, you can use the following special tokens to insert dynamic values:
 - {{pageNumber}} - Will be replaced with the current page number.
 - {{pageCount}} - Will be replaced with the total number of pages.
 - opacity (Number) Opacity of the watermark. 1.0 is completely opaque, 0.0 is completely transparent. Default is 1.0.
 - color (String) Text color. Can be any valid CSS color name (like "red") or hex value (like "#FF0000"). Default is "black".
 - fontFamily (String) Font family for the text. Default for SVG output is to use the browser's default font. Default for raster output is unspecified.
 - fontSize (String) Font size specified in points (like "12pt"). Default for SVG output is to use the browser's default font size. Default for raster output is unspecified.
 - fontWeight (String) Determines the font weight. Possible values:
 - "normal" (default)
 - "bold"
 - fontStyle (String) Determines the font style. Possible values:
 - "normal" (default)
 - "italic"
 - textDecoration (String) Possible values:
 - "none" (default)
 - "underlined"
 - slope (String) Controls the text angle. Default is "up". Possible values:
 - "up" - Text will start in the lower-left corner of the page and extend upwards to the upper-right corner of the page. (default)
 - "down" - Text will start in the upper-left corner of the page and extend downwards to the lower-right corner of the page.
 - *image watermark:*
 - type: "image" (String) **Required.** Must be set to "image" to indicate the object represents an image watermark.
 - opacity (Number) Opacity of the watermark. 1.0 is completely opaque, 0.0 is completely transparent. Default is 1.0.
 - src (String) **Required.** URL or work file id of a PNG image to use for this watermark. When using a URL, the URL must be accessible from the server where PrizmDoc Server is running. > **NOTE:** The src MUST be a PNG. If you use a different image format, invalid watermarks will be created.
 - horizontalAlign (String) Determines the horizontal position of the watermark. Default is "center". Possible values:
 - "left" - Image will be horizontally anchored near the left side of the page.
 - "center" Image will be horizontally anchored to the center of the page. (default)
 - "right" - Image will be horizontally anchored near the right side of the page.
 - verticalAlign (String) Determines the vertical position of the watermark. Default is "middle". Possible values:
 - "top" - Image will be vertically anchored near the top of the page.
 - "middle" Image will be vertically anchored to the middle of the page. (default)
 - "bottom" - Image will be vertically anchored near the bottom of the page.
 - scale (Number) - Determines the relative size of the image as compared to the size of the page. Value must be between 0.0 and 1.0. A value of 1.0 indicates the image will be scaled to the size of the page while 0.0 indicates the image will be scaled infinitesimally small and will not be rendered. Default is 0.25.
 - autoSize (String) When set, the image will be automatically sized to fill the page (any value provided for scale, horizontalAlign, and verticalAlign will be ignored). Possible values:
 - "fit" - Image will be scaled to be as large as possible while still completely fitting within the page.



The aspect ratio of the image is maintained.



- `"fill"` - Image will be scaled to be large enough that the entire page is covered by the image. Some of the image may fall off the edge of the page, but the entire page is guaranteed to be covered by some part of the image. The aspect ratio of the image is maintained.
- `"stretch"` - Image width and height will be independently resized so that the image width and height are the same as the page. The aspect ratio of the image is ignored.
- `watermarkText` (String) *Deprecated*. Providing this value no longer has any effect. Use `watermarks` instead.
- `pageContentEncryption` (String) - Controls whether or not page content will be encrypted for the viewing session. See the [Enabling Content Encryption](#) topic for more information about this feature. Possible values:
 - `"default"` - Product configuration will be used to determine whether or not page content will be encrypted (see `viewing.contentEncryption.enabled` in the [Central Configuration](#) file).
 - `"enabled"` - Page content will be encrypted for the viewing session.
 - `"disabled"` - Page content will not be encrypted for the viewing session.
- `countOfInitialPages` (Integer) Number of pages which should be eagerly converted, or `0` if all pages should be eagerly converted. Default is `0`.
- `startConverting` (String) When the `documentSource` is `"http"` or `"file"`, controls whether initial pages should be converted as soon as the document has been acquired. Default is `"none"`. Possible values:
 - `"none"` - Conversion will begin only after the session is explicitly started or page content or attributes are requested. (default)
 - `"initialPages"` - Conversion will begin as soon as the source document has been acquired.
- `contentType` (String) - Determines what kind of browser content will be eagerly pre-generated (other kinds of content may still be generated if explicitly requested). Possible values:
 - `"svgb"` - Pre-generate fully-optimized SVG (uses a unicode inline font to store glyph definitions). Smallest possible SVG, but may not be compatible with some browsers. Recommended whenever possible.
 - `"svga"` - Pre-generate partially-optimized SVG (uses a non-unicode inline font to store only the most frequently-occurring glyph definitions). May not be compatible with some browsers. Use only if `"svgb"` content is not compatible with the target browser.
 - `"svg"` - Pre-generate unoptimized SVG (no font is used; glyph definitions are expressed as SVG path operations). Broadest compatibility with browsers but typically *much* larger, so it renders and scrolls much slower than `"svgb"` and `"svga"`. Not recommended. Use only as a fallback if both `svgb` and `svga` are not compatible with the target browser, or the use of webfonts is disabled in the target browser.
 - `"png"` - Pre-generate raster content.
- `serverCaching` (String) Controls whether output is kept for potential reuse by other viewing sessions. Default is `"full"`. Possible values:
 - `"full"` - Output will be written to disk on the server and retained for reuse by other viewing sessions created for the same source document. Output will not be deleted until the configured viewing cache lifetime is reached (which is a full day with an out-of-box configuration; see `viewing.cacheLifetime` in [Central Configuration](#)). Saves processing time if a source document is viewed repeatedly before the cached data is deleted, but does consume more disk space. (default)
 - `"none"` - Output will be written to disk on the server but only retained for the duration of the viewing session and never shared with other viewing sessions. Once the viewing session expires, the output will be deleted from the disk. Saves disk space if you know that it is unlikely a source document will ever be viewed more than once, but can result in redundant processing if the same source document is viewed repeatedly.
- `serverSideSearch` (String) Determines whether the server-side search feature will be available for the viewing session. Default is `"enabled"`. Possible values:
 - `"enabled"` - Server-side search will be available for the viewing session. (default)
 - `"disabled"` - Server-side search will not be available for the viewing session.
- `attachmentIndex` (Integer) - *Intended for use only by PrizmDoc Server when it automatically creates viewing sessions for attachments. This is not a property your application should use.* If the source document is an attachment that belongs to another document (such as an email), the 1-based index of this attachment in the list of all attachments (e.g. `1` means it was the first attachment, `2` means it was the second, etc.) or `0` to indicate that the source document is not an attachment. Default is `0`.
- `attachmentDisplayName` (String) - *Intended for use only by PrizmDoc Server when it automatically creates viewing sessions for attachments. This is not a property your application should use.* If the source document is an attachment that belongs to another document (such as an email), the filename of the attachment or `null`. Default is `null`.

Successful Response

Response Body

JSON with metadata about the created viewing session.

- `viewingSessionId` (String) Unique id for this viewing session.
- `affinityToken` (String) Affinity token for this viewing session. Present when clustering is enabled.

Error Responses

Some error responses will have a JSON body with an `errorCode` and `errorDetails`:

Status Code	JSON <code>errorCode</code>	Description
480	"MissingInput"	A required input value was not provided. See <code>errorDetails</code> in the response body.
480	"InvalidInput"	An invalid input value was used. See <code>errorDetails</code> in the response body.
480	"InputNotSupportedWithServerCacheEnabled"	Can occur when <code>minSecondsAvailable</code> is specified when <code>serverCaching</code> is not set to "none". See <code>errorDetails</code> in the response body.

Some error responses will include an error message in an `Accusoft-Status-Message` header.

Example

Request

```
POST prizmdoc_server_base_url/PCCIS/V1/ViewingSession
Content-Type: application/json

{
  "tenantId": "my application name",
  "externalId": "my-unique-document-name.docx",
  "render": {
    "html5": {
      "alwaysUseRaster": false
    }
  },
  "minSecondsAvailable": 1500,
  "watermarks": [
    {
      "type": "text",
      "opacity": 0.6,
      "text": "jdoe\n67.79.169.114\n11/13/2014 2:24 PM\nNOT FOR DISTRIBUTION",
      "color": "red",
      "fontFamily": "Consolas",
      "fontSize": "16pt",
      "fontWeight": "bold",
      "verticalAlign": "bottom",
      "horizontalAlign": "right"
    }
  ]
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8

{
```

```

"viewingSessionId": "GcIsIsEGbLV2_V9yy4NzmK2HB-JuLOH--
A9sZ16cla9tx00ZDBGfqlG4kKu0r_GyEps4wWCvDwn4dpnZAR76Uw"
"affinityToken": " S2ZqtGi9vUAXBgdmM/PNNpCM4CApe9NxLIp/4QnAHlg="
}

```

GET /PCCIS/V1/ViewingSession/u{viewingSessionId}

Returns the metadata associated with a valid, active viewing session. The properties returned will be those provided in the POST request that created the viewing session.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The viewingSessionId which identifies the viewing session.

Successful Response

JSON metadata about the viewing session:

- `creationTime` (String) A UTC-formatted time string representing the moment the viewing session was created.
- `minSecondsAvailable` (Integer) Minimum number of seconds the viewing session will exist.
- `documentSource` (String) Will be one of the following:
 - `null` - No value was specified in the initial POST to create the viewing session. Same as `"api"`.
 - `"api"` - The source document will be provided by the calling application via an API request.
 - `"http"` - PrizmDoc Server will download the source document from the URL specified by `externalId`.
 - `"file"` - PrizmDoc Server will use a local file as the source document using the file path specified by `externalId`.
- `externalId` (String) Indicates where the source document was acquired from when the `documentSource` is `"http"` or `"file"`:
 - *When documentSource is "http"*: The URL where the source document was acquired from
 - *When documentSource is "file"*: The path to the file on the server which was used as the source document
- `documentExtension` (String) - File extension of the source document provided in the original POST to create the viewing session or `null` if no value was provided.
- `password` (String) Password provided to open the source document.
- `tenantId` (String) Custom tenant id assigned to the viewing session in the original POST.
- `origin` (Object) Custom origin data included in the original POST to create the viewing session.
- `render` (Object) Rendering options in use for the viewing session:
 - `html5` (Object) HTML5 output options in use for the viewing session:
 - `alwaysUseRaster` (Boolean) Indicates whether or not only raster data, instead of SVG, will be created for the viewing session.
 - `svgMaxImageSize` (Number) The maximum edge length, in pixels, that is allowable for any image when creating the SVG. See [POST /ViewingSession](#) for more info.
 - `vectorTolerance` (Number) For CAD documents, the amount of path simplification that is allowable when creating the SVG. See [POST /ViewingSession](#) for more info.
 - `rasterResolution` (Integer) *Deprecated and unused. Provided only for backwards compatibility.*
 - `flash` (Object) *Deprecated and unused. Provided only for backwards compatibility.*
 - `optimizationLevel` (Integer) *Deprecated and unused. Provided only for backwards compatibility.*
- `watermarks` (Array of Objects) Objects describing watermarks which should be applied to page content. See [POST /ViewingSession](#) for more info.
- `watermarkText` (String) *Deprecated and unused. Provided only for backwards compatibility.*
- `pageContentEncryption` (String) - Will be one of the following:
 - `null` - No value was specified in the initial POST to create the viewing session. Same as `"default"`.
 - `"default"` - Product configuration will be used to determine whether or not page content will be encrypted (see `viewing.contentEncryption.enabled` in the [Central Configuration](#) file).
 - `"enabled"` - Page content will be encrypted for the viewing session.
 - `"disabled"` - Page content will not be encrypted for the viewing session.

- `countOfInitialPages` (Integer) Number of pages which will be eagerly converted, or 0 if all pages should be eagerly converted. Default is 0.
- `startConverting` (String) When the `documentSource` is "http" or "file", indicates whether initial pages will be converted as soon as the document has been acquired. Default is "none". Possible values:
 - `null` - No value was specified in the initial POST to create the viewing session. Same as "none".
 - `"none"` - Conversion will begin only after the session is explicitly started or page content or attributes are requested. (default)
 - `"initialPages"` - Conversion will begin as soon as the source document has been acquired.
- `contentType` (String) - Indicates what kind of browser content will be eagerly pre-generated (other kinds of content may still be generated if explicitly requested). Will be one of the following:
 - `null` - No value was specified in the initial POST to create the viewing session.
 - `"svgb"` - Pre-generate fully-optimized SVG (uses a unicode inline font to store glyph definitions). Smallest possible SVG, but may not be compatible with some browsers. Recommended whenever possible.
 - `"svga"` - Pre-generate partially-optimized SVG (uses a non-unicode inline font to store only the most frequently-occurring glyph definitions). May not be compatible with some browsers. Use only if "svgb" content is not compatible with the target browser.
 - `"svg"` - Pre-generate unoptimized SVG (no font is used; glyph definitions are expressed as SVG path operations). Broadest compatibility with browsers but typically *much* larger, so it renders and scrolls much slower than "svgb" and "svga". Not recommended. Use only as a fallback if both `svgb` and `svga` are not compatible with the target browser, or the use of webfonts is disabled in the target browser.
 - `"png"` - Pre-generate raster content.
- `serverCaching` (String) Indicates whether output is kept for potential reuse by other viewing sessions. Will be one of the following:
 - `"full"` - Output will be written to disk on the server and retained for reuse by other viewing sessions created for the same source document. Output will not be deleted until the configured viewing cache lifetime is reached (which is a full day with an out-of-box configuration; see `viewing.cacheLifetime` in [Central Configuration](#)). Saves processing time if a source document is viewed repeatedly before the cached data is deleted, but does consume more disk space. (default)
 - `"none"` - Output will be written to disk on the server but only retained for the duration of the viewing session and never shared with other viewing sessions. Once the viewing session expires, the output will be deleted from the disk. Saves disk space if you know that it is unlikely a source document will ever be viewed more than once, but can result in redundant processing if the same source document is viewed repeatedly.
- `attachmentIndex` (Integer) - If the viewing session was created for an attachment on a parent source document (such as an email), the 1-based index of this attachment in the list of all attachments (e.g. 1 means it was the first attachment, 2 means it was the second, etc.). 0 otherwise.
- `attachmentDisplayName` (String) - If the viewing session was created for an attachment on a parent source document (such as an email), the filename of the attachment this viewing session was created for. `null` otherwise.
- `serverSideSearch` (String) Indicates whether or not the server-side search feature will be available for the viewing session. Will be one of the following:
 - `"enabled"` - Server-side search is available for the viewing session. (default)
 - `"disabled"` - Server-side search is not available for the viewing session.

Error Responses

Status Code	Reason Phrase	Description
403	The session is invalid or has expired	You requested a valid <code>{viewingSessionId}</code> but it is no longer available.
500	Internal Server Error	Can occur if you forget to prefix the <code>{viewingSessionId}</code> portion of the URL with <code>u</code> , or if you simply request an invalid <code>{viewingSessionId}</code> .

Example

Request

```
GET prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uXYZ...
```


Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "origin": {},
  "render": {
    "flash": {
      "optimizationLevel": 1
    },
    "html5": {
      "alwaysUseRaster": false,
      "rasterResolution": 150
    }
  },
  "password": null,
  "watermarkText": null,
  "externalId": null,
  "attachmentIndex": 0,
  "attachmentDisplayName": null,
  "tenantId": null,
  "creationTime": "2015-10-14T11:55:32.6521255Z",
  "countOfInitialPages": 0,
  "documentSource": null,
  "documentExtension": "help",
  "serverCaching": "full",
  "startConverting": null,
  "contentType": null,
  "pageContentEncryption": null,
  "watermarks": []
}
```

PUT /PCCIS/V1/ViewingSession/u{viewingSessionId}/SourceFile?FileExtension={FileExtension}

Uploads the source document to be viewed.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The viewingSessionId which identifies the viewing session.
{FileExtension}	Required for text-based formats (such as <code>txt</code>, <code>csv</code>, <code>html</code>). File extension of the document being uploaded. This parameter may or may not be required depending on the file type and whether Format Detection is enabled. Note that the extension must not include the leading period (for example, <code>csv</code> is accepted but <code>.csv</code> will return an error). Extensions are not case sensitive. > NOTE: <i>If Format Detection is disabled, then <code>FileExtension</code> is always required. When provided, <code>FileExtension</code> may only include alpha-numeric characters.</i>

If Format Detection is enabled (the default), the use of `FileExtension` is as follows:

- If we can auto-detect the file format, we ignore `FileExtension`.
- If we cannot auto-detect the file format, we require a `FileExtension` be specified, otherwise we return HTTP 580 with an `errorCode` of `"UnrecognizedFileFormat"`. This most-commonly occurs with text-based source documents (such as `txt`, `csv`, or `html`).

Request Body

The bytes of the source document.

Successful Response

A simple HTTP 200 status code indicating the file was received.

Error Responses

Some error responses will have a JSON body with an `errorCode` and `errorDetails`:

Status Code	JSON <code>errorCode</code>	Description
480	"IncorrectUsage"	For any new viewing session, you can give it either 1) a single <i>source</i> document for viewing or 2) two documents (<i>original</i> and <i>revised</i>) which should be viewed as a comparison, but you cannot do both. If you receive this error from this URL, it is because one or more files (<i>original</i> and/or <i>revised</i>) have already been provided for comparison.

Some error responses will include an error message in an `Accusoft-Status-Message` header:

Status Code	<code>Accusoft-Status-Message</code> Header	Description
580	"Unrecognized File Format"	Occurs when the file format cannot be automatically detected. To avoid this, provide an explicit <code>FileExtension</code> query string parameter for this kind of file. This most-commonly occurs with text-based source documents where we require an explicit <code>FileExtension</code> to know the file format (such as <code>txt</code> , <code>csv</code> , or <code>html</code>). You may need to create a new viewing session for the <code>PUT</code> to succeed.
580	"Watermark image download failed"	Occurs when image watermark defined in the Viewing Session cannot be downloaded.

Example

Request

```
PUT prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uXYZ.../SourceFile?FileExtension=doc
<<file bytes>>
```

Response

```
HTTP/1.1 200 OK
```

PUT `/v2/viewingSessions/{viewingSessionId}/sourceFile/original`

Used when viewing a comparison of two documents, uploads the first of the two documents, the *original* document.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.

Request Body

The bytes of the *original* document (for comparison with a *revised* document).

Successful Response

A simple HTTP 200 status code indicating the file was received.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No viewing session with the provided <code>viewingSessionId</code> could be found.
480	"CannotChangeDocument"	The viewing session already has an <i>original</i> document assigned.
480	"UnsupportedFormatForComparison"	The uploaded file was not a Word document (for comparison viewing, we currently only support ".doc" and ".docx" files).
480	"IncorrectUsage"	For any new viewing session, you can give it either 1) a single <i>source</i> document for viewing or 2) two documents (<i>original</i> and <i>revised</i>) which should be viewed as a comparison, but you cannot do both. If you receive this error from this URL, it is because a single <i>source</i> document has already been provided.
480	"FeatureNotLicensed"	The server's license does not allow the use of the MSO (Microsoft Office) feature, so document comparison is not possible.
480	"FeatureDisabled"	The server has not been configured to allow the use of the Microsoft Office renderer, so document comparison is not possible.

Example

Request

```
PUT prizmdoc_server_base_url/v2/viewingSessions/XYZ.../sourceFile/original
<<file bytes>>
```

Response

```
HTTP/1.1 200 OK
```

PUT /v2/viewingSessions/{viewingSessionId}/sourceFile/revised

Used when viewing a comparison of two documents, uploads the second of the two documents, the *revised* document.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.

Request Body

The bytes of the *revised* document (for comparison with an *original* document).

Successful Response

A simple HTTP 200 status code indicating the file was received.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No viewing session with the provided <code>viewingSessionId</code> could be found.
480	"MissingInput"	A required input value was not provided. See <code>errorDetails</code> in the response body.
480	"InvalidInput"	An invalid input value was used. See <code>errorDetails</code> in the response body.
480	"CannotChangeDocument"	The viewing session already has a <i>revised</i> source document assigned.
480	"UnsupportedFormatForComparison"	The uploaded file was not a Word document (for comparison viewing, we currently only support "doc" and "docx" files).
480	"IncorrectUsage"	For any new viewing session, you can give it either 1) a single <i>source</i> document for viewing or 2) two documents (<i>original</i> and <i>revised</i>) which should be viewed as a comparison, but you cannot do both. If you receive this error from this URL, it is because a single <i>source</i> document has already been provided.
480	"FeatureNotLicensed"	The server's license does not allow the use of the MSO (Microsoft Office) feature, so document comparison is not possible.
480	"FeatureDisabled"	The server has not been configured to allow the use of the Microsoft Office renderer, so document comparison is not possible.

Example

Request

```
PUT prizmdoc_server_base_url/v2/viewingSessions/XYZ.../sourceFile/revised
<<file bytes>>
```

Response

```
HTTP/1.1 200 OK
```

PUT /PCCIS/V1/ViewingSession/u{viewingSessionId}/SourceRef

Attach an existing [work file](#) (or, when doing comparison, two work files) as the source document to be used for a viewing session. Using a source reference can be particularly useful when you want to avoid repeatedly uploading the same source file to the back end.

Request

Request Headers

Name	Description
Content-Type	Must be <code>application/json</code>

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.

Request Body when assigning reference for a single document

- `refType` (string) **Required.** Must be set to `"workFile"`.
- `fileId` (string) **Required.** The id of the work file to use as the source document.

Request Body when comparing two documents

- `refType` (string) **Required.** Must be set to `"comparison"`.
- `original`
 - `refType` (string) **Required.** Must be set to `"workFile"`.
 - `fileId` (string) **Required.** The id of the work file to use as the original document.
- `revised`
 - `refType` (string) **Required.** Must be set to `"workFile"`.
 - `fileId` (string) **Required.** The id of the work file to use as the revised document.

Successful Response

A simple HTTP 200 status code.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
480	"MissingInput"	A required input value was not provided. See <code>errorDetails</code> in the response body.
480	"InvalidInput"	An invalid input value was used. See <code>errorDetails</code> in the response body.
480	"NotFound"	The given work file has expired or does not exist. See <code>errorDetails</code> in the response body.
480	"CannotChangeDocument"	The viewing session already has a source document assigned. See <code>errorDetails</code> in the response body.
480	"UnsupportedFormatForComparison"	One of the files provided for comparison was not a Word document (for comparison viewing, we currently only support "doc" and "docx" files). See <code>errorDetails</code> in the response body.
480	"IncorrectUsage"	For any new viewing session, you can give it either 1) a single <i>source</i> document for viewing or 2) two documents (<i>original</i> and <i>revised</i>) which should be viewed as a comparison, but you cannot do both. If you have already provided a <i>source</i> document and then attempt to provide an <i>original</i> and <i>revised</i> document pair, or vice versa, you will receive this error.
480	"FeatureNotLicensed"	The server's license does not allow the use of the MSO (Microsoft Office) feature, so document comparison is not possible.
480	"FeatureDisabled"	The server has not been configured to allow the use of the Microsoft Office renderer, so document comparison is not possible.

Examples

Assigning a single source document

```
PUT prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uXYZ.../SourceRef
Content-Type: application/json

{
  "refType": "workFile",
  "fileId": "CVBuD7DbQYNoJDqByGierQ",
}
```

```
HTTP/1.1 200 OK
```

Assigning two documents to be viewed as a comparison

```
PUT prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uXYZ.../SourceRef
Content-Type: application/json

{
  "refType": "comparison",
  "original": {
    "refType": "workFile",
    "fileId": "CVBuD7DbQYNoJDqByGierQ",
  },
  "revised": {
    "refType": "workFile",
    "fileId": "5qTYa3gzN9gYUb5SzqUhqg",
  }
}
```

```
HTTP/1.1 200 OK
```

GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/SourceFile? ContentDispositionFilename={ContentDispositionFilename}

Gets the source document in use for a viewing session.

Request**URL Parameters**

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.
{ContentDispositionFilename}	The filename as a URL-encoded string, without extension, to be used in the <code>Content-Disposition</code> response header (appropriate file extension such as <code>doc</code> or <code>docx</code> will automatically be added). By default, the value will be <code>SourceFile.<ext></code> .

Response Headers

Name	Description
<code>Content-Disposition</code>	Indicates to a browser that the response body should be treated as a file download and specifies the filename the browser should use.

Name	Description
Content-Type	The most-specific MIME type for the returned document or <code>application/octet-stream</code> otherwise.

Response Body

The raw bytes of the viewing session's source document.

Example

Request

```
GET prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uXYZ.../SourceFile?
ContentDispositionFilename=MonthlySalesReport
```

Response

```
200 OK
Content-Type: application/msword
Content-Disposition: attachment; filename=MonthlySalesReport.docx; filename*=UTF-8'MonthlySalesReport.docx

<<file bytes>>
```

GET /v2/viewingSessions/{viewingSessionId}/sourceFile/original?contentDispositionFilename={contentDispositionFilename}

When viewing a comparison of two documents, gets the *original* document used in the comparison. The document returned will be an identical copy of the document originally provided.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.
{contentDispositionFilename}	The filename as a URL-encoded string, without extension, to be used in the <code>Content-Disposition</code> response header (appropriate file extension such as <code>doc</code> or <code>docx</code> will automatically be added). By default, the value will be <code>OriginalSourceFile.<ext></code> .

Response Headers

Name	Description
Content-Disposition	Indicates to a browser that the response body should be treated as a file download and specifies the filename the browser should use.
Content-Type	The most-specific MIME type for the returned document or <code>application/octet-stream</code> otherwise.

Response Body

The raw bytes of the first of the two documents being viewed as a comparison, the *original* document.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No viewing session with the provided <code>viewingSessionId</code> could be found.
480	"DocumentNotProvidedYet"	An <i>original</i> document has not been provided to the viewing session yet.
480	"IncorrectUsage"	For any new viewing session, you can give it either 1) a single <i>source</i> document for viewing or 2) two documents (<i>original</i> and <i>revised</i>) which should be viewed as a comparison, but you cannot do both. If you receive this error from this URL, it is because a single <i>source</i> document was provided for viewing, so there will never be an <i>original</i> comparison document to get.

Example

Request

```
GET prizmdoc_server_base_url/v2/viewingSessions/XYZ.../sourceFile/original?
contentDispositionFilename=OldMonthlySalesReport
```

Response

```
200 OK
Content-Type: application/msword
Content-Disposition: attachment; filename=OldMonthlySalesReport.docx; filename*=UTF-8'OldMonthlySalesReport.docx

<<file bytes>>
```

NOTE: See [GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/SourceFile](#) for an example when filename includes non-ASCII characters.

GET /v2/viewingSessions/{viewingSessionId}/sourceFile/revised? contentDispositionFilename={contentDispositionFilename}

When viewing a comparison of two documents, gets the *revised* document used in the comparison. The document returned will be an identical copy of the document originally provided.

The response will set the `Content-Type` header to the most-specific MIME type it can or `application/octet-stream` otherwise.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.
{contentDispositionFilename}	The filename as a URL-encoded string, without extension, to be used in the <code>Content-Disposition</code> response header (appropriate file extension such as <code>doc</code> or <code>docx</code> will automatically be added). By default, the value will be <code>RevisedSourceFile.<ext></code> .

Response Headers

Name	Description
Content-Disposition	Indicates to a browser that the response body should be treated as a file download and specifies the filename the browser should use.
Content-Type	The most-specific MIME type for the returned document or <code>application/octet-stream</code> otherwise.

Response Body

The raw bytes of the second of the two documents being viewed as a comparison, the *revised* document.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No viewing session with the provided <code>viewingSessionId</code> could be found.
480	"DocumentNotProvidedYet"	A "revised" document has not been associated with the viewing session yet.
480	"IncorrectUsage"	For any new viewing session, you can give it either 1) a single <i>source</i> document for viewing or 2) two documents (<i>original</i> and <i>revised</i>) which should be viewed as a comparison, but you cannot do both. If you receive this error from this URL, it is because a single <i>source</i> document was provided for viewing, so there will never be a <i>revised</i> comparison document to get.

Example

Request

```
GET prizmdoc_server_base_url/v2/viewingSessions/XYZ.../sourceFile/revised?
contentDispositionFilename=NewMonthlySalesReport
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/msword
Content-Disposition: attachment; filename=NewMonthlySalesReport.docx; filename*=UTF-8'NewMonthlySalesReport.docx

<<file bytes>>
```

NOTE: See [GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/SourceFile](#) for an example when filename includes non-ASCII characters.

GET /v2/viewingSessions/{viewingSessionId}/restrictions

Returns information about any restrictions enforced by the server for the current viewing session.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.

Successful Response

JSON with information about any restrictions currently in place for the viewing session:

- `delayEnabled` (Boolean) - Indicates whether the server has enforced an artificial delay before the document conversion results are allowed to be delivered to the viewer.
- `delaySecondsRemaining` (Integer) - The number of seconds remaining in the artificially-imposed delay before the document conversion results are allowed to be delivered to the viewer. Only present when `delayEnabled` is `true`.
- `downloadDisabled` (Boolean) - Indicates whether downloading of the source document has been disabled for this viewing session.
- `markupBurnersDisabled` (Boolean) Indicates whether the markup burner feature has been disabled for this viewing session.
- `formInfoDisabled` (Boolean) Indicates whether the form detection feature has been disabled for this viewing session.

Example Responses

When the product is licensed and the form detection feature is available

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "delayEnabled": false,
  "downloadDisabled": false,
  "markupBurnersDisabled": false,
  "formInfoDisabled": false
}
```

When the product is unlicensed (evaluation mode), and 9 seconds of initial delay are remaining:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "delayEnabled": true,
  "delaySecondsRemaining": 9,
  "downloadDisabled": true,
  "markupBurnersDisabled": true,
  "formInfoDisabled": true
}
```

GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/FileId

Gets the [work file](#) id of the source document in use for a viewing session.

Regardless of how the source document was provided or acquired, internally PrizmDoc Server will always ensure a work file exists for the source document. This URL allows you to get the work file id for a viewing session's source document. This can be helpful if you want to create a new viewing session using the same source document; you can simply create the new viewing session and then attach the existing work file to it as the source document (see [PUT /SourceRef](#)).

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.

Successful Response

JSON with the work file id.

- `fileId` (String) - Work file id for the source document of the viewing session.
- `affinityToken` (String) Work file affinity token. Present when clustering is enabled.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
480	"DocumentNotProvidedYet"	Occurs when you create a viewing session with a <code>documentSource</code> of "api" but have not yet made the API call(s) necessary to provide the source document.
480	"LicenseCouldNotBeVerified"	The server's license could not be verified. If you are evaluating the product without a license, the product is running in evaluation mode and this particular part of the product is unavailable without a license. If you have a license, make sure you configured your license correctly, that your license has not expired, and that you have not exceeded any license limits (such as, for a Cloud License, the total number of logical CPU cores in use).

Example

Request

```
GET prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uXYZ.../FileId
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8

{
  "fileId": "10GayKfK2dGtC8jhOtSknw",
  "affinityToken": "ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

POST

/PCCIS/V1/ViewingSession/u{viewingSessionId}/Notification/SessionStarted

Ensures PrizmDoc Server has started the process of converting a viewing session's source document to HTML.

The `User-Agent` header should be set to an appropriate browser string to enable the PrizmDoc Server to begin generating the correct content for the Viewer.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.

Request Headers

Name	Description
<code>Content-Type</code>	Should be <code>application/json</code> .

Name	Description
User-Agent	Browser user agent string which PrizmDoc Server uses to generate the most-appropriate content for the viewer.

Request Body

- `viewer` (String) - Type of viewer being used. Default is "HTML5". Possible values:
 - "HTML5"

Successful Response

A simple HTTP 200 status code indicating the session has been started.

Error Responses

Status Code	Description
500	Can occur if your request body is not JSON. Make sure your JSON contains at least an empty object ({}).

Example

Request

```
POST prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uXYZ.../Notification/SessionStarted
Content-Type: application/json

{
}
```

Response

```
HTTP/1.1 200 OK
```

POST

/PCCIS/V1/ViewingSession/u{viewingSessionId}/Notification/SessionStopped

Invalidates a viewing session so that it can no longer be used.

Invalidating a viewing session can be useful when:

1. A viewing session has been created but something prevents your application from being able to provide a source document for the viewing session.
2. An end user has finished using a viewing session and no additional access to the viewing session should be allowed.

When you invalidate a viewing session, you must provide a JSON body which specifies what HTTP status code and reason phrase should be returned as an error for future requests to the invalidated viewing session. This allows you to control what sort of HTTP response other people or applications will receive if they make requests to your invalidated viewing session.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.

Request Headers

Name	Description
Content-Type	Should be application/json.

Request Body

- `httpStatus` (Integer) - HTTP status code to respond with for any future requests to this viewing session. Default is 580.
- `endUserMessage` (String) - Error message to use for both the HTTP reason phrase and `Accusoft-Status-Message` response header for any future requests to this viewing session. Default is "Session is stopped".
- `accusoftErrorNumber` (Integer) - Value for the `Accusoft-Status-Number` response header for any future requests to this viewing session. Default is 580 or `httpStatus` if it is provided.
- `serverLogMessage` (String) - Message that should be emitted to the PrizmDoc Server log file when the session is stopped. Default is "The viewing session is stopped on request from the client."

Successful Response

A simple HTTP 200 status code indicating the session has been stopped.

Example

Request

```
POST prizmdoc\_server\_base\_url/PCCIS/V1/ViewingSession/uXYZ.../Notification/SessionStopped
Content-Type: application/json

{
  "httpStatus": 418,
  "endUserMessage": "My custom end user error message"
}
```

Response

```
HTTP/1.1 200 OK
```

Then, if any future requests are made to the viewing session, the response will be:

```
HTTP/1.1 418 My custom end user error message
Accusoft-Status-Message: My custom end user error message
Accusoft-Status-Number: 418
```

POST

[/PCCIS/V1/ViewingSession/u{viewingSessionId}/Notification/SessionErrored](#)

Mark a viewing session as errored so that it can no longer be used but it is still possible to request the source file. (See [GET /PCCIS/V1/ViewingSession/u{viewingSessionId}/SourceFile](#))

Erroring a viewing session can be useful when:

1. It is necessary to stop its usage but keep a source document available to download.
2. The `password` is invalid for a password-protected source document so the viewing session can be replaced using [POST /PCCIS/V1/ViewingSession/u{viewingSessionId}/Replacement](#) API.

When you mark a viewing session as errored, you must provide a JSON body which specifies what HTTP status code and reason phrase should be returned as an error for future requests to the invalidated viewing session. This allows you to control what sort of

HTTP response other people or applications will receive if they make requests to your errored viewing session.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.

Request Headers

Name	Description
Content-Type	Should be <code>application/json</code> .

Request Body

- `httpStatus` (Integer) - HTTP status code to respond with for any future requests to this viewing session. Default is 580.
- `endUserMessage` (String) - Error message to use for both the HTTP reason phrase and `Accusoft-Status-Message` response header for any future requests to this viewing session. Default is "Session is errored".
- `accusoftErrorNumber` (Integer) - Value for the `Accusoft-Status-Number` response header for any future requests to this viewing session. Default is 580 or `httpStatus` if it is provided.
- `serverLogMessage` (String) - Message that should be emitted to the PrizmDoc Server log file when the session is stopped. Default is "The viewing session is errored on request from the client."

Successful Response

A simple HTTP 200 status code indicating the session has been errored.

Example

Request

```
POST prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uXYZ.../Notification/SessionErrored
Content-Type: application/json

{
  "httpStatus": 418,
  "endUserMessage": "My custom end user error message"
}
```

Response

```
HTTP/1.1 200 OK
```

Then, if any future requests are made to the viewing session, the response will be:

```
HTTP/1.1 418 My custom end user error message
Accusoft-Status-Message: My custom end user error message
Accusoft-Status-Number: 418
```

POST /PCCIS/V1/ViewingSession/u{viewingSessionId}/Replacement

Replace the existing viewing session with a new one that has a new `password` parameter value. The other original viewing session

parameters are preserved. If a source document is uploaded it is attached to the newly created viewing session with a new password. The old viewing session is stopped.

The replacement API is useful when a session is errored because of either an invalid or missing password, and you want to replace it with a new session with the same parameters against the same document but with a modified password.

Request

Request Headers

Name	Description
Content-Type	Should be <code>application/json</code>

Request Body

- `password` (String) Password to use when opening a password-protected source document. If the parameter is not provided a new viewing session is created with a default null `password` parameter value.

URL Parameters

Parameter	Description
<code>{viewingSessionId}</code>	The <code>viewingSessionId</code> which identifies the viewing session.

Successful Response

Response Body

JSON with metadata about the created viewing session.

- `viewingSessionId` (String) Unique id for this viewing session.

Error Responses

Status Code	Reason Phrase	Description
403	The session is invalid or has expired	You requested a valid <code>{viewingSessionId}</code> but it is no longer available.
500	Internal Server Error	Can occur if you forget to prefix the <code>{viewingSessionId}</code> portion of the URL with <code>u</code> , or if you simply request an invalid <code>{viewingSessionId}</code> .

Some error responses will have a JSON body with an `errorCode` and `errorDetails`:

Status Code	JSON <code>errorCode</code>	Description
480	"PropertyNotReplaceable"	Unsupported property to replace was used. See <code>errorDetails</code> in the response body.

Example

Request

```
POST prizmdoc_server_base_url/PCCIS/V1/ViewingSession/uXYZ.../Replacement
Content-Type: application/json

{
  "password": "123"
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8

{
  "viewingSessionId": "THMMytF4ACHrxmN2bXj4vahx4Gnwly_kEeFt20XtRau-
z43pPHIjUy5JXJ05Wj1MaqvdfsXp98JxIk7ALWkukg"
}
```

DELETE /PCCIS/V1/ViewingSession/u{viewingSessionId}

Deletes a viewing session. Only possible for viewing sessions created with `serverCaching` set to `"none"`.

Request

URL Parameters

Parameter	Description
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session.

Successful Response

HTTP 204 indicating the session was deleted.

Error Responses

Status Code	JSON <code>errorCode</code>	Description
404	-	No viewing session with the provided {viewingSessionId} could be found.
580	"CannotDeleteCachedViewingSession"	The viewing session you attempted to delete was not created with <code>serverCaching</code> set to <code>"none"</code> .

Unsupported Routes

These unsupported REST APIs are only for internal use by Accusoft products and components:

- [Unsupported Routes](#)

Unsupported Routes

Introduction

The following endpoints are intended only for internal use by Accusoft products and components.

If you are a PrizmDoc Server administrator, you should expose these endpoints to ensure that all product functionality is available.

However, if you are an application developer, you should not call these endpoints. They are undocumented,

unsupported, and subject to change without notice.

Available URLs

- POST /unsupported/excelShapeExtractors
- GET /unsupported/excelShapeExtractors/{processId}

PrizmDoc Server .NET SDK

The [PrizmDoc Server .NET SDK](#) is a wrapper around the PrizmDoc Server REST APIs, making it easy to use PrizmDoc Server functionality in .NET. It is an [open source library](#) published as a [NuGet package](#).

To learn more and get started, visit the PrizmDoc Server .NET SDK documentation site:

<https://help.accusoft.com/PrizmDoc/sdks/server/dotnet/v1/>

Troubleshooting

This Troubleshooting section is designed to help you work through some common issues.

If you're experiencing an issue that isn't covered in this section, make sure to also visit the [Technical FAQs for PrizmDoc Viewer](#) on our website. These FAQs are updated regularly, based on Customer Support interactions. Also be sure to check our [Release Notes](#).

This section contains the following information:

- [Document Viewing Issues](#)
 - [Office Files](#)
 - [Text Files](#)
 - [CAD/PDF Files](#)
- [Viewing Package Issues](#)
- [Redaction & Annotation Issues](#)
- [Form Field Detector Error Messages](#)
- [Log File Growth](#)
- [Memory Consumption Issues](#)
- [PrizmDoc Server Health Issues](#)

See also [Accusoft Support](#).

Document Viewing Issues

Introduction

This section outlines possible reasons why some documents may render differently than expected:

- [Office Files](#)
- [Text Files](#)
- [CAD/PDF Files](#)

Note for all Formats

Viewing pages in documents which have more than 10,000 pages

PrizmDoc Viewer currently **does not support** viewing pages past the 10,000th page.

As a possible workaround, consider splitting the original document into a number of smaller PDF files. For example, use 1,000 pages per document for viewing. You can use the PrizmDoc [Content Conversion Service API](#) for splitting an original document into smaller files. Specify a PDF as the output format and specify the desired page range for each resulting PDF.

Related FAQs

- [Why do the fonts in my document look different when rendered in PrizmDoc?](#)
- [My document appears to be loading incorrectly. Are there any troubleshooting steps I can take?](#)
- [Why is it taking a long time to load PDF documents when viewing a document in PrizmDoc using Google Chrome 71?](#)
- [In PrizmDoc, why is my document appearing smaller on the page relative to the viewer?](#)

Office Files

Introduction

This section outlines possible reasons why some Office documents may be rendered differently than expected.

MS Office Documents do not Render in the Viewer using MSO Rendering Mode

Below are some reasons for intermittent or continuous issues when viewing MS Office documents.

1. Check the environment (CPU/RAM) to ensure it is properly sized according to the [PrizmDoc Server Sizing](#) recommendations. If you do not meet the sizing requirements, increase your system resources to match sizing recommendations.
2. Check the `NonInteractiveSystemHeapSize` value using the table in the [Registry Changes](#) section. If it is not correct, set the registry key to match the table and restart your server.
3. Verify whether your MS Office version is up to date. You can find the latest available updates for your version by checking: [Update history for Office 2013](#) or [Update history for Office 2016 C2R](#). If you are not on the latest version, upgrade to the latest patch for your version and restart the server.

Office Documents Render Differently in PrizmDoc than MS Office

PrizmDoc has two different rendering engines: LibreOffice (LO mode) and MS Office (MSO mode).

In LO rendering mode, it is expected that there will be some differences from MS Office. To make sure the PrizmDoc is operating in MSO rendering mode, check whether your [PrizmDoc Viewer license](#) includes MSO support and whether the steps described in [Natively Render MSO Documents](#) are followed.

Alternatively, check how the document is rendered in MS Office Print Preview mode, and compare it with PrizmDoc output.

For Excel documents, please note that PrizmDoc has its own pagination mechanism enabled by default. For details, refer to the [Central Configuration](#) topic.

If you prefer the MSO rendering and are not currently licensed, please speak with your sales representative for [information regarding this feature](#).

Excel Documents Render in PrizmDoc with a lot of Pages Compared to MS Excel

When comparing PrizmDoc Viewer output to MS Excel output, consider that MS Excel shows the document content in "Normal View" mode by default, which does not split the content into pages. Your document might have special formatting or small text in a cell that is not visible on the screen.

To see how many pages are actually there, switch to **Page Layout View** mode or go to **Print Preview** -> **Print Entire Workbook** and see how many pages there are, and what the document will look like when printed. Once you've located the cell with the unexpected data, you can delete it, and then it will render the same.

Office Documents Render Asian Characters Incorrectly

When opening Office documents with Asian characters in PrizmDoc Viewer, the document may contain unrecognizable characters.

This can occur when the specific font type in the document is not available on the PrizmDoc Server. The PrizmDoc Server will attempt to use the most suitable font available which can produce less accurate results.

In order to assign a proper match of fonts, refer to the following topic for more information: [Substitute Fonts for Office Rendering Fidelity](#)

Related FAQs

- [Why am I unable to see pictures on certain Excel spreadsheets in PrizmDoc Viewer?](#)
- [In PrizmDoc, why do I fail to load/convert Excel documents with the error "Exception from HRESULT: 0x800AC472"?](#)

Text Files

Introduction

This section outlines possible reasons why some text documents rendered differently than expected.

The text file shows only a single line of text when rendered by PrizmDoc

For text files which only show a single line of text, the text file may have Mac-style line endings - CR (or '\r') only without line feed character LF (or '\n'). To verify, open text file in Notepad++ and select View, Show Symbol, Show All characters.

CAD/PDF Files

Introduction

This section outlines possible reasons why some CAD/PDF documents rendered differently than expected.

Related FAQ

- [Why are the fonts in my CAD files showing up garbled/unrecognizable/not as expected?](#)

Viewing Package Issues

Introduction

This section outlines how to troubleshoot viewing package failures.

Viewing Package Creation Failures

If you are getting Viewing Package creation failures, make sure you are not hitting the file system or database capacity limits for the viewing package storage.

Redaction & Annotation Issues

Introduction

This section outlines possible reasons why some documents may have been redacted differently than expected.

Redacting the Attached PDF File does not Remove Certain Text

While we do our best to make sure the markup burner API handles PDF documents appropriately, due to complexity of the PDF specification and a variety of PDF producers, there might be some rare PDF documents causing this problem. To guarantee secure content redaction, we suggest the following approach:

1. Perform your normal redaction including redaction reasons, if necessary.
2. Convert the output PDF to TIFF using the [Content Converter API](#).
3. Convert the TIFF back to PDF using the same [Content Converter API](#), using the option to create a searchable PDF to allow for searching the redaction reasons and other text content in the output PDF.

Redactions are Unexpectedly Applied on Subsequent Pages

The reason for the issue is that the PDF document can contain one image that is shared between several pages. PrizmDoc not only redacts the text, but also the underlying raster image. As a result, if the image (e.g., a background image) is shared between multiple pages, all of those pages get the same update, such that the redaction rectangle will start to render on all pages for which that source image is shared. This behavior is similar to Adobe Acrobat redaction, with the difference that PrizmDoc uses black redactions by default, while the Acrobat uses white. This product behavior is by design. You can specify a non-default redaction color using the [/v2/redactionCreators API](#).

Note: This behavior only occurs when downloading a PDF with redactions actually burned in (Redactions > Redact); it does not occur when using "See Through" redactions.

When downloading a PDF with Chinese text annotations, the PDF does not properly display the Chinese characters.

When the PrizmDoc Viewer sends a text annotation markup to burn it in, it includes the font parameters (including the font name) for the annotation text to use. The supported out-of-the-box list of fonts is defined in the viewer sample template, and these fonts do not have Chinese characters. So, the PrizmDoc backend can't find the Chinese unicodes for the provided font, and these characters are replaced with the square (tofu) characters.

To resolve this issue, you can customize the viewer sample template code to add a font with Chinese characters to make sure that PrizmDoc markup burner API can process the Chinese annotation text, assuming that this Asian font is actually available on the server system.

To update the template to add new font:

Customize **C:\Prizm\Samples\dotnet\webforms\full-viewer-sample** to add a new font (`FangSong`) as follows:

1. Make sure that the font with Chinese characters (e.g., `FangSong`) is available on the server (it may need to be installed separately).
2. Open the html template **viewer-assets\src\templates\contextMenuTemplate.html** and find the following code:

```
    <code></code>
```

```
<div data-pcc-toggle-id="dropdown-font" class="pcc-dropdown">
  <div><%= fontArial %></div>
  <div><%= fontComicSans %></div>
  <div><%= fontCourier %></div>
  <div><%= fontCourierNew %></div>
  <div><%= fontGeneva %></div>
  <div><%= fontGeorgia %></div>
  <div><%= fontHelvetica %></div>
  <div><%= fontTimes %></div>
  <div><%= fontTimesNewRoman %></div>
  <div><%= fontVerdana %></div>
</div>
```

3. Insert the following line:

```
<div><%= fontFangSong %></div>
```

Note: *The first font in the list is the default font.*

4. Save the file.
5. Open the file **viewer-assets\src\languages\en-US.json** and find the code:

```
"fontArial": "Arial",
"fontBoldText": "Bold Text",
"fontColorMenuItem": "Font Color",
"fontComicSans": "Comic Sans MS",
"fontCourier": "Courier",
"fontCourierNew": "Courier New",
```

6. Add the line:

```
"fontFangSong": "FangSong",
```

7. Save the file.
8. Follow the instructions from **C:\Prizm\Samples\dotnet\webforms\full-viewer-sample\viewer-assets\README.md** to build the sample.

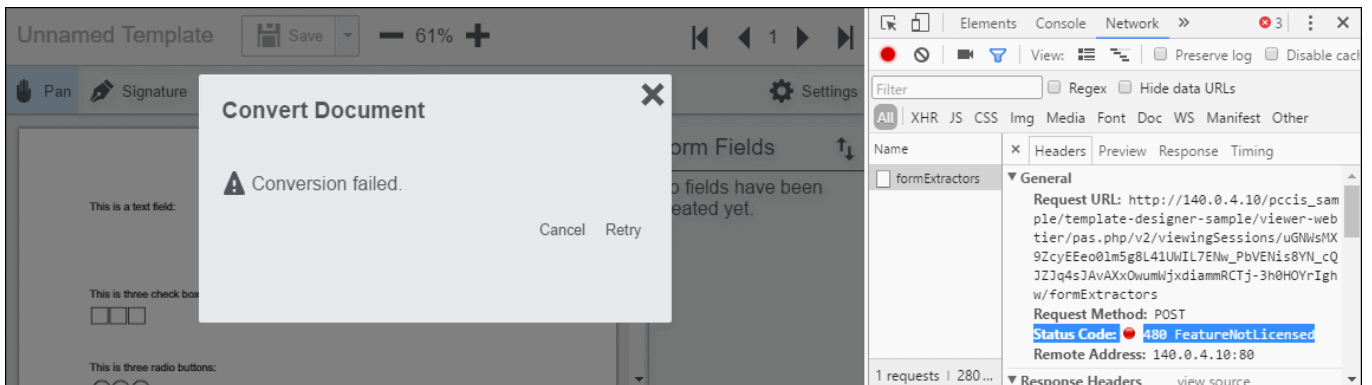
After reloading the PrizmDoc Viewer web page, a new font will be available in the Viewer and can be selected to be explicitly used by the markup burner to burn in the Chinese text.

Form Field Detector Error Messages

Introduction

This section outlines how to troubleshoot Form Field Detection errors.

Receiving a "Conversion Failed" Error when Trying to Convert Form Fields in the Template Designer Sample



This issue may occur if the feature is not enabled. To verify if the feature is enabled, check the following:

1. Open the **Developer Tools (F12)** in your browser and select the **Network** tab.
2. Attempt to convert the form fields again for the same document.
3. Look for Status Code: `480 FeatureNotLicensed`.

If you want to enable the feature, refer to [Feature Licensing](#) for more details.

If you don't want to enable the feature, click **Continue Without Converting** the next time the dialog appears prompting you to convert the document.

Log File Growth

Introduction

This section outlines possible reasons why some log files grow larger than expected.

The `plb.sep_single.log.x/plb.sep_multi.log.x` and `ContentConversionService.log.x` logs are taking up excessive space

If those log files occupy some 50% or more of the total log size, it could indicate that you are polling `/v2/contentConverters` too frequently. To verify how often you are polling the post request you can check the following:

1. Search for lines matching "taskBegin.GET.v2/contentConverters" in those log files to see how frequently it is polled.
2. Compare the number of POST contentConverters (lines matching "taskBegin.POST.v2/contentConverters") vs the number of GET contentConverters (lines matching "taskBegin.GET.v2/contentConverters") to see how many poll requests are issued against each conversion, on average.

When polling "GET /v2/contentConverters", frequently the API call executes constantly with no pause in between each call. This would cause the ratio of number of polls per POST to be high depending on how long it takes to convert the document. Modifying the polling to check once every 3-5 seconds would reduce the number of logged items significantly and reduce the size of the log file.

Related FAQs

- [PrizmDoc logs have timestamps, what timezone are they in?](#)

Memory Consumption Issues

Introduction

This section outlines how to troubleshoot memory consumption issues that may occur with the PrizmDoc Server.

Java Services use an Unexpected Amount of Memory

When PrizmDoc Server is under a heavy processing load (specifically when performing redaction and burn-in operations with the documents being processed), it is possible for the PrizmDoc Java backend services to unexpectedly grow memory usage constantly. When this type of unexpected memory consumption occurs, the OOM Killer on Linux could kill the Java backend service (or it could cause Windows to run out of memory).

There is a dedicated JVM option, "-XX:MaxRAM", to limit the memory usage by the whole Java process. Using this option helps to avoid situations when the Java process grows more and more. It limits unmanaged and managed heaps very effectively for Linux and Windows platforms. This option is especially applicable to applications running in Linux containers.

Note, if you start using the option "-XX:MaxRAM", the default value for the managed heap size will likely be smaller than recommended. To get the maximum effectiveness and performance for PrizmDoc we recommend setting the Java heap size explicitly. There is the option "-XX:MaxRAMPercentage" to set a relative Java heap size and the parameter "-Xmx" or "-XX:MaxHeapSize". All can be used to limit the managed heap size. For more information read the Java article [here](#).

Currently there are two settings in the central configuration file, one per Java service: PDF Processing (PDFPS) and Email Processing (EPS) services - "pdfps.jvm.opts" and "eps.jvm.opts" (see [Central Configuration](#)). It can be used to tune the default behavior of Java services.

For the PDF Processing service you can set the following options for JVM to prevent memory consumption issues:

```
pdfps.jvm.opts: "-XX:MaxRAM=<value1> -XX:MaxRAMPercentage=<value2> "
```

or

```
pdfps.jvm.opts: "-XX:MaxRAM=<value1> -Xmx<value2> "
```

Where the "value1" is the maximum amount of memory that the JVM may use for the Java heap before applying ergonomics heuristics. The default value is the maximum amount of available memory to the JVM process. The "value2" in percent (or byte size for Xmx) is the maximum amount of memory that the JVM may use for the Java heap before applying ergonomics heuristics as a percentage of the maximum amount determined as described in the -XX:MaxRAM option.

For example, to reliably limit the PDF Processing service maximal memory usage to 25% of available memory on the instance with 32 GB RAM, you could use:

```
pdfps.jvm.opts: "-XX:MaxRAM=8G -XX:MaxRAMPercentage=75.0"
```


or

```
pdfps.jvm.opts: "-XX:MaxRAM=8G -Xmx6G"
```

For this case, the whole Java heap will be limited by 8 GB. The managed heap size will be limited to 75% of all available heap size of 8G. In testing, we found that relations of 75% for managed heap and 25 % for unmanaged show a good balance between performance and stability for PrizmDoc Viewer.

This is not a strict recommendation though; you should determine the exact value by experimenting and observing the impact of PrizmDoc on your environment. Note that this value impacts product performance. A higher value means better performance in the case of intensive operations with PDF files because it allows you to put more documents in memory, work with them at the same time, and not call the garbage collector frequently.

PrizmDoc Server Health Issues

Introduction

This section outlines possible reasons why the PrizmDoc Server may not show as healthy.

If you've [checked the PrizmDoc Server health](#), and any of the services are unhealthy, try the following:

- Check your permissions, login, and password of the service user
- Verify that you are using a valid license

If your services are still unhealthy, there are some common reasons for this:

- **The System has Limited Resources**
- **Linux-specific Instructions**
 - [The cache is corrupted or inaccessible](#)
 - [PAS or PrizmDoc Server isn't running](#)
- **Windows-specific Instructions**
 - [The cache is corrupted or inaccessible](#)
 - [PAS or PrizmDoc Server isn't running](#)
 - [PAS isn't connected to PrizmDoc Server](#)

The System Has Limited Resources

The system resources for a machine running PrizmDoc are described in [Sizing Servers](#). Please note that PrizmDoc will require at least a few gigabytes free while running and no conversions are taking place. If your system has PrizmDoc installed and idles with 1-2GB free RAM, then the services will run into stability issues. For more information, refer to the [Memory Consumption Issues](#) topic.

Linux-specific Instructions

The Cache is Corrupted or Inaccessible

A common problem is not running or installing PrizmDoc Server as the root user, which is a requirement for running PrizmDoc on a Linux server. Correcting this will solve permission-related issues while accessing the cache. To clear a potentially corrupted cache manually, use the following steps:

1. Run: `/usr/share/prizm/scripts/pccis.sh stop`
2. Delete the contents of: `/usr/share/prizm/cache/`
3. Start up PrizmDoc Server: `/usr/share/prizm/scripts/pccis.sh start`

PAS or PrizmDoc Server is Not Running

Another common issue is that either PrizmDoc Application Services (PAS) or PrizmDoc Server isn't running.

To check that PAS is running:

1. Navigate to: `/usr/share/prizm/pas/pm2`
2. Run: `./pas.sh status`

To check that the PrizmDoc Server is running:

1. Navigate to: `/usr/share/prizm/scripts`
2. Run: `./pccis.sh status`

If the steps above have been checked and the Admin page shows that PrizmDoc is unlicensed, then the license key will need to be verified by Accusoft Support at support@accusoft.com. You can find the current license key in `/usr/share/prizm/prizm-services-config.yml`:

```
license.solutionName: PrizmDoc12
license.key: 2.0.E6Aorkh8...
```

In the example above, the license key is "2.0.E6Aorkh8..." To locate your license key, copy everything after "license.key:" and send your **full license key** to Accusoft Support.

NOTE: Your solutionName and license key may differ from the example above.

If everything seems to be functioning, and a file is still not displayed in the Viewer, there may be an issue with the conversion of the specific file. If this is the case, send the file to support@accusoft.com for evaluation and submission to our engineering team.

Windows-specific Instructions

The Cache is Inaccessible or Corrupted

You will need to clear the cache. To clear the cache manually, use the following steps:

1. Stop the **Windows PrizmDoc Server**.
2. Delete the contents of **C:\Prizm\Cache**.
3. Start up the **Windows PrizmDoc Server**, and restart **IIS**.

If the Admin page shows that PrizmDoc is unlicensed, then the license key will need to be verified by Accusoft Support at support@accusoft.com. You can find the current license key in `C:\Prizm\prizm-services-config.yml`:

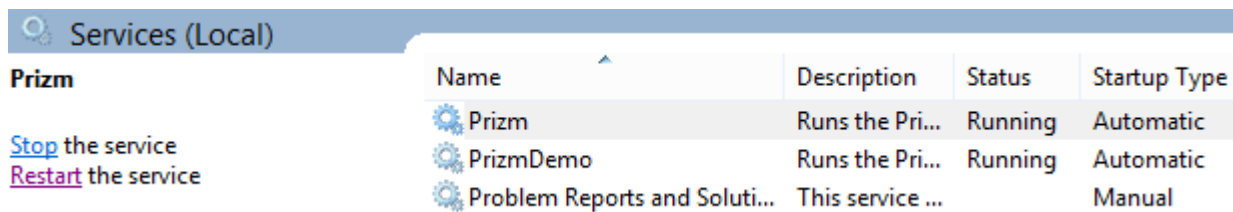
```
5 license.solutionName: PCC10
6 license.key: 2.0.EAWgWs2X4Aeb3Fes8sU1fA4FfVjYTKfAUA
7
```

Your license key will be everything after the equal sign on **Line 6**. Copy and send your **full key** to Accusoft Support.

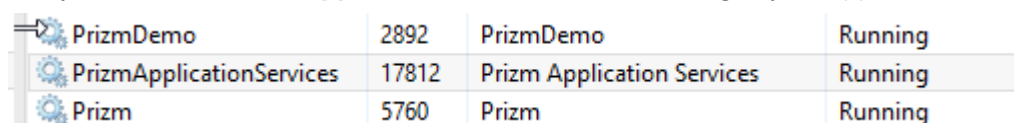
PrizmDoc Server or PAS is Not Running

If you've [checked the PrizmDoc Server health](#), and any of the services are unhealthy, try the following:

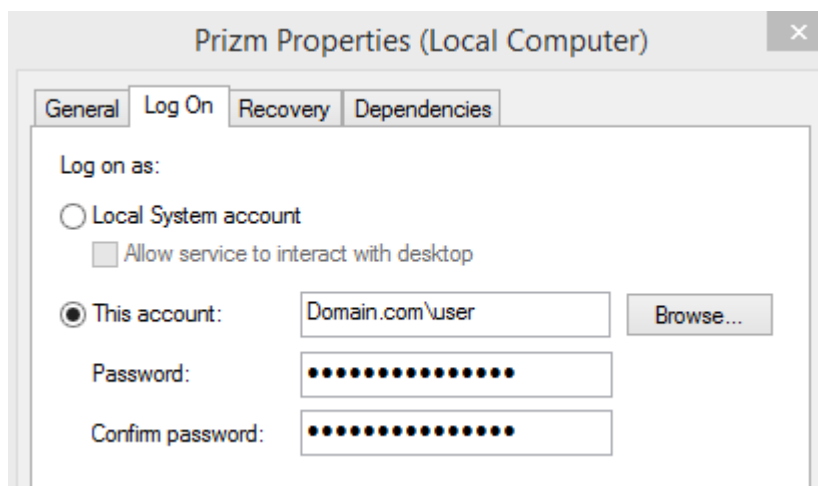
1. Verify that the **PrizmDoc Server** is running:



2. Verify that the **PrizmDoc Application Service (PAS)** is running (if your application is using PAS):

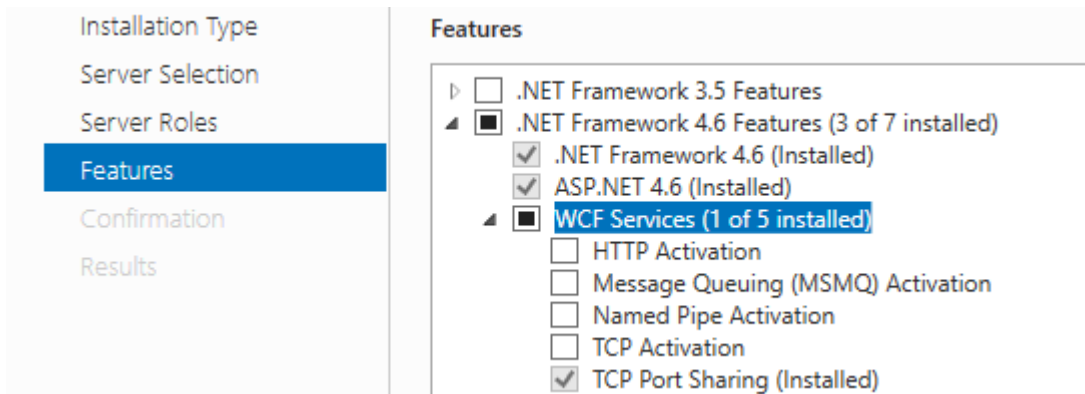


3. Verify that the "Log On As" user for the PrizmDoc Server has not recently had a password change. Right-click **Properties** and re-enter the **user password**:

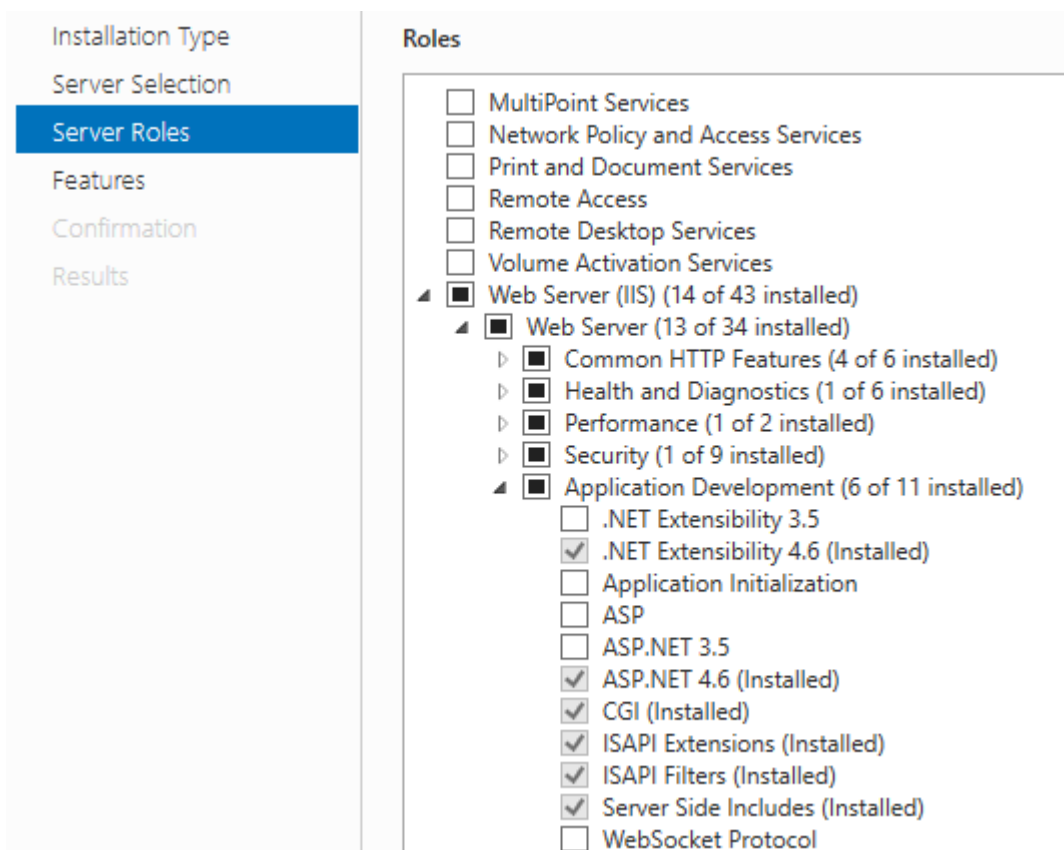


4. Navigate to **C:\ProgramData\Accusoft** and make sure that the user from Step 2 above has full permissions to the folders under **Prizm**:
 - a. Right-click the **Prizm folder** and select **Properties**.
 - b. Navigate to the **Security Tab**.
 - c. Select **Advanced**.
 - d. Select **Change permissions** (if required).
 - e. Make sure that the user from Step 2 above has **Full Control**.
 - f. Be sure to select **Replace all child object permission entries with inheritable permission entries from this object** to avoid discrepancy.
 - g. Apply any changes that were made.
5. Open the **Internet Information Services (IIS) Manager**.
6. Check that the **PrizmDoc Service Web Site** is running with its application pool.
 - a. The actions menu on the right-hand side will show if it is running.
 - b. Click on **Basic Settings** in the action menu to see the application pool set.

- c. After verifying the **pool name**, go to **Application Pools**.
- d. Verify that the **application pool** is started.
- e. If the Application pool identity is a **user**, and that user is different from the one checked for the Windows PrizmDoc Service, verify that the password and permissions are set in the same way as Step 3 above.
- f. Open the **Windows Features** and check that **ASP.NET Framework 4.5+** is currently installed:



- 7. Also, open **Server Roles** and check that **ASP.NET 4.5+** is enabled under **Web Server (IIS)**:



- 8. Check the **health** of the PrizmDoc Server at:

<http://localhost:18681/admin>

Your license status will be displayed at the top.

The health of the services will be displayed (which will display the same data in JSON format).

PAS is Not Connected to PrizmDoc Server

To check that PAS can connect to PrizmDoc Server, this API call:

```
GET http://localhost:3000/servicesConnection
```

Returns the status of PAS connectivity to the PrizmDoc Server, whether local or configured through Accusoft Services.

Successful Response

```
`200 OK`
```

```
`OK`
```

This response shows that the connection to PrizmDoc Server is successful, but does not take into account whether or not those services are healthy. If you need to check the health of the services, please make a call to them directly.

Error Responses

```
`580`
```

The response shows that PAS is not properly configured to communicate with PrizmDoc Server. For help with configuring PAS to communicate with PrizmDoc Server, please review [Configuring the PrizmDoc Server Connection](#)

If everything seems to be functioning, and a file is still not displayed in the Viewer, there may be an issue with the conversion of the specific file. If this is the case, send the file to support@accusoft.com for evaluation and submission to our engineering team.

Related FAQs

- [Why is my Prizm Services status "unhealthy" and showing a clock for the mongo-manager-service?](#)
- [Why is the PrizmDoc service health page showing unhealthy when MSO is enabled?](#)
- [When running PrizmDoc on Linux, why is the service health page showing unhealthy when MSO is enabled?](#)

1 Index

- 1. Integrating the Viewer, 97-99
- 2. Choosing a Backend & Creating a Viewing Session, 99-103
- 3. Setting up a Reverse Proxy, 103-119
- Accusoft Policy on Log Changes, 401
- Accusoft Support, 8-9
- Add a Custom Button, 149-151
- Add Custom Image Stamps, 147-148
- Add Keyboard Shortcuts, 151-154
- Adjust Caching Parameters, 358-361
- Administrator Guide, 292-295
- Affinity Tokens & Cluster Mode, 369-372
- API Data Types, 877-879
- API Reference, 405
- Application Development, 882 , 972
- Architecture & Design, 123-126
- Architecture Basics, 129-131
- Architecture Overview, 93-94
- Attachments, 921-922 , 1049-1051
- Authenticating Requests, 872-873
- Back-end Proxy, 879-882
- Base URL for PAS, 876-877
- Base URL for PrizmDoc Server , 971-972
- Build a Custom User Interface , 154-158
- Build the E-Signature Viewers, 206-207
- CAD/PDF Files, 1092
- Central Configuration, 349-354
- Change Annotation Default Values, 176
- Change Encryption Keys for Public use Token Generation, 361-362
- Change the Position of the Menu Bar, 158-159
- Check PrizmDoc Server Health, 327-328
- Check the Connection to PrizmDoc Server, 388-389
- Class: AjaxResponse, 441-443
- Class: BurnRequest, 443-451
- Class: Comment, 451-461
- Class: Conversation, 461-469
- Class: ConversionRequest, 469-478
- Class: DocumentHyperlink, 478-482

Class: Error, 482-483
Class: ESigner, 807-811
Class: Event, 483-486
Class: ImageStamps, 486-489
Class: LoadMarkupLayersRequest, 489-495
Class: Mark, 495-573
Class: MarkupLayer, 573-592
Class: MarkupLayerCollection, 592-599
Class: MouseTool, 599-606
Class: ObservableCollection, 606-609
Class: PrintRequest, 609-614
Class: Promise, 614-618
Class: Revision, 618-621
Class: RevisionsRequest, 621-625
Class: SearchRequest, 625-631
Class: SearchResult, 631-639
Class: SearchTask, 639-642
Class: SearchTaskResult, 642-647
Class: Signature Control, 647-650
Class: SignatureDisplay, 650-651
Class: TemplateDesigner, 811-814
Class: ThumbnailControl, 651-658
Class: Viewer, 658-659
Class: ViewerControl, 659-762
Cloud Authentication, 872
Cloud License (Deprecated), 334-340
Cluster Management, 1046-1049
Clustering, 365-368 , 395
Compare Documents, 251-252
Compare Documents with PAS, 214-216
Configuration Options, 133
Configure a Cluster, 330-331
Configure Microsoft Office Conversion Connectivity, 362-363
Configure the Comments Panel, 135-136
Configure the E-Signature Viewers, 204-206
Configure the Viewer, 132-133
Configuring, 389 , 349
Content Conversion Demo, 264
Content Conversion Service, 972-983

- Content Converters, 922-926**
- Content Converters (Deprecated), 926-929**
- Convert Content with Content Conversion Service, 252-264**
- Copyright Information, 13**
- Create a Custom Mouse Tool, 159-160**
- Create a Custom Tab, 160**
- Customize the Book Reader Viewer, 207-210**
- Customize the E-Signature Viewers, 202**
- Customize the Markup, 160-162**
- Customize the Mouse Tools, 162-165**
- Customize the Styles, 165-167**
- Define the View Mode, 136-137**
- Definitions, 9-11**
- Design Basics, 126-129**
- Developer Guide, 120-121**
- Digital Rights Management Configuration, 137-138**
- Disable the Print Button, 167-168**
- Document Rendering Specifics, 302-303**
- Document Viewing Issues, 1090**
- Enable Content Encryption, 138-141**
- Enable Multiple Redaction Reasons, 168-171**
- Error Reporting, 398-401**
- E-Signature Controls, 802-803**
- External: jQuery, 410**
- External: jQuery.fn, 803-807**
- Fill in Fields Programmatically, 207**
- Form Definitions, 929-933**
- Form Extractors, 933-943 , 1051-1057**
- Form Field Detector Error Messages, 1094-1095**
- General Information, 876 , 971**
- Getting Started, 90**
- Glossary, 9**
- Handle Specific Routes with PAS, 212-214**
- Health, 919-920**
- Health Status, 1042-1046**
- How & When to use CORS, 395**
- How to Configure the Demo on Linux, 266-268**
- How to Configure the Demo on Windows, 264-266**

- How to Customize the Viewer, 148-149**
- How To Examples, 250-251**
- How to Use Pre-defined Search, 142-145**
- HTML5 Viewing, 1057-1065**
- Illustrating the Viewing Sequence, 94-97**
- Image Stamps, 943-945**
- Implement Caching Strategies, 355-358**
- Implement our Top Features, 122**
- Initial Integration, 93**
- Initialization Parameters, 133**
- Install Asian Fonts on Traditional Linux Install Packages, 321-322**
- Install on Windows, 306 , 308-310 , 381-382**
- Install Using Traditional Linux Install Packages, 316-321**
- Install Using Traditional Linux Install Packages on a Headless Environment, 322-326**
- Installing, 303 , 377**
- Installing with Traditional Linux Install Packages (deprecated), 314 , 384-386**
- Integrate PrizmDoc Viewer Releases with Your Code , 131-132**
- Legacy ASP.NET MVC Sample, 184-188**
- Legacy ASP.NET WebForms Sample, 188-195**
- Legacy Create Session, 945-947**
- Legacy JSP Sample, 195-201**
- Legacy Samples, 183**
- Legacy Viewers, 202**
- Legal, 13**
- Licensing, 331-332**
- Linux, 397 , 372-374**
- Load Annotations from the Web Tier, 176-177**
- Localize the Viewer, 145-146**
- Log File Growth, 1095-1096**
- Markup Burner XML Specification, 228-238**
- Markup Burners, 947-951 , 983-989**
- Markup JSON Specification, 238-250**
- Markup Layers, 951-957**
- Markup XML, 957-960**
- Memory Consumption Issues, 1096-1097**
- Metered License, 332-334**
- Migrate from PrizmDoc Cloud Servers to PrizmDoc Viewer Self-Hosted Servers, 268-269**
- Mixin: Data, 762-765**
- Mixin: SessionData, 765-768**

Modify viewer.js, 182-183

Module: button-set, 814-817

Module: checkbox-collection, 817-820

Module: data-persist, 820

Module: date-picker, 820-822

Module: download-signed-form, 823-824

Module: download-signed-form-trigger, 822-823

Module: dropdown, 824-827

Module: event-store, 827-837

Module: field-edit, 837-838

Module: field-list, 838-839

Module: fill-checklist, 839-840

Module: fill-form-controller, 840-842

Module: fill-main-toolbar, 842-843

Module: fill-progress, 843-844

Module: form-controller, 844-845

Module: form-extraction, 845-846

Module: form-summary, 846-847

Module: form-tools, 847-849

Module: global-settings-menu, 849-850

Module: global-settings-trigger, 850-851

Module: keyboard-controller, 851-852

Module: multiple-selection, 852-853

Module: notification, 853-854

Module: page-navigation, 854-855

Module: profile-manager, 855-856

Module: state-store, 856-864

Module: svg-icons, 864-865

Module: template-io, 865-866

Module: template-manager, 866-867

Module: template-name-header, 867-868

Module: text-input, 869-871

Module: zoom-fit, 871-872

Namespace: Ajax, 768-775

Namespace: fn, 411-425

Namespace: Language, 775-777

Namespace: MarkSchema, 788-801

Namespace: MarkupLayerSchema, 801-802

Namespace: MouseTools, 777-780
Namespace: PCCViewer, 425-441
Namespace: Signatures, 780-781
Namespace: Util, 781-788
Natively Render MSO Documents, 311-312
New Terms, 11-13
Node-Locked License (Deprecated), 340-349
OAuth, 873-875
OEM License, 334
Office Files, 1090-1092
Optimize Cache Performance for Cluster Environments, 395-396
Optimize Cache Performance for Cluster Mode, 368-369
Overview , 1-3
Packaging Log Files for Support, 404
PAS, 210
PAS Configuration, 389-394
PAS Database Administration & Maintenance, 394-395
PAS REST API, 875-876
PCCIS Configuration, 354-355
Perform Auto-Redaction, 269-273
Plain Text Redactors, 989-996
Pre-Convert Documents, 216-219
Pre-Populate Fields in the E-Signature Viewer, 219-221
PrizmDoc Application Services, 375
PrizmDoc Cells Overview, 8
PrizmDoc Server, 295-296 , 224-225
PrizmDoc Server .NET SDK, 1089
PrizmDoc Server Health Issues, 1097-1101
PrizmDoc Server REST API, 970-971
Redaction & Annotation Issues, 1092-1094
Redaction Creators, 996-1008
Registry Changes, 310-311
Release Notes, 59-89
Reorganize Menus, 171-173
Requirements & Supported Environments for Traditional Linux Install Packages, 314-316 , 386-387
Run PAS on Clusters, 396-397
Sample Applications, 91-92
Scroll the Viewer Programmatically, 173-174
Search Contexts, 1008-1025

Search Tasks, 960-970 , 1025-1033

Search Tips, 401-403

Security Guidance, 298-302

Self-Hosted Administration, 919 , 1042

Server Sizing, 375-377

Server Sizing , 296-298

Set the Initial Zoom Factor, 174-175

Set up a Viewing Session for a CAD Drawing which has XREF Dependencies, 273-279

Set up Your Database for use with PAS, 210-212

Software License Agreement, 13-17

Start & Stop PrizmDoc Server, 372

Starting & Stopping, 397

Subscribe to Events, 175-176

Substitute Fonts for Office Rendering Fidelity, 363-364

Supported File Formats, 3-7

Text Files, 1092

Third-Party Attributions, 17-58

Troubleshooting, 1090

Try It!, 90-91

uiElements, 133

Unattended Install & Uninstall, 312-314 , 383-384

Uninstall PrizmDoc Server on Windows, 314

Uninstall Traditional Linux Install Packages, 326-327 , 387

Unsupported Routes, 1088 , 1088-1089

Upgrade from Legacy Configuration, 364-365

Upgrade PrizmDoc Viewer, 328-330

Use a Custom Resource Path, 146-147

Use a Viewing Session, 280-283

Use Pre-loaded Search Parameters, 133-135

Use the Markup JSON Schema, 279-280

Use the PrizmDoc Server API, 225-228

Using Docker, 303-306 , 377-381

Viewer, 123

Viewer Control, 405-410

Viewer Modular Design, 202-204

Viewer Requirements, 7-8

Viewer Support, 920-921 , 1049

Viewing Package Creators, 904-915

Viewing Package Issues, 1092

Viewing Packages, 915-919

Viewing Sessions, 882-904 , 1065-1088

Watermark Content in a Viewing Session, 283-291

Windows, 397-398 , 374-375

Windows Requirements & Supported Environments, 306-308 , 382-383

Work Effectively with Large Documents, 122-123

Work Files, 1033-1042

Work with Annotation Layers, 177-178

Work with Annotations, 176

Work with Annotations Programmatically, 178-181

Work with Document Comparison Programmatically, 181-182

Work with Viewing Packages, 221-224