

## PrizmDoc PDF



This PDF file is an offline version of the PrizmDoc online help file and may not contain the most up-to-date content. To verify that you're using the latest version of the documentation, please go to <http://www.accusoft.com/products/prizmdoc/documentation/>.

- [About PrizmDoc](#)
- [Get Started with PrizmDoc](#)
- [Administrator Guide](#)
- [Developer Guide](#)
- [End User Guides](#)
- [API Reference](#)

## About PrizmDoc

PrizmDoc™ is a suite of HTTP services and JavaScript browser code that makes it easy for you to add document viewing, redaction, annotation, conversion, and other document processing services to your web site or applications using only HTML5 standard technologies. Users can view a variety of document types, including Microsoft Office, Adobe PDF, AutoCAD, and many other **document and image types** without having to install native applications.

The key features of PrizmDoc include:

- [True Zero-Footprint Viewing](#)
- [Support for the Documents You Care About](#)
- [Full Markup/Annotation Capabilities](#)
- [Full Redaction Capability](#)
- [Large Document Viewing & Server-Side Search](#)
- [Advanced Search](#)
- [Customization](#)
- [Pre-Convert Your Documents for Even Faster Viewing](#)
- [Add-on Licensed Options for Self-Hosted](#)
- [About PrizmDoc Help](#)

### [True Zero-Footprint Viewing](#)

If you're building an application that requires easy-to-implement document viewing, using PrizmDoc to render and display documents is perfect for you. By converting any document and image into web standard formats, you can display documents right in a web browser on any desktop or mobile device. With just a few lines of code, you can embed PrizmDoc's customizable document viewer into your web site, application or document management solution. No installation is required on your customer's device, because everything they need is delivered directly in the browser - no downloads required!

### [Support for the Documents You Care About](#)

Accusoft's extensive history with document and image file formats guarantees that your customers will see accurate, consistent representations of the files they care about.

### [Full Markup/Annotation Capabilities](#)

PrizmDoc's annotation layering functionality allows multiple users to mark up a document and to see and comment on each other's additions. Keeping all marks in a single document with the ability to show or hide individual user contributions provides clear and concise version control.

### [Full Redaction Capability](#)

Download redacted versions of sensitive, original documents. PrizmDoc's redaction technology securely removes any trace of the redacted items from the final document. Redactions can be applied manually or automatically via search results or pre-populated search terms. Automatic redaction can also be set using regular expressions to search and redact format-specific content, such as Social Security numbers, phone numbers, emails, and dates.

## Large Document Viewing & Server-Side Search

Added in v12.0, Large Document Viewing & Server-Side Search allows you to view and search extremely large documents in no time. The product supports large documents in a variety of formats, providing faster rendering and superior performance for documents with 1000+ pages. The Large Document Viewing & Server-Side Search feature reduces the memory load in the Viewer, minimizing the load time and optimizing performance for viewing and searching large documents. For example, PrizmDoc loads a document the size of *War and Peace* very quickly and is ready to view and search in seconds, rather than minutes. For more details on this feature, refer to [Working Effectively with Large Documents](#).

## Advanced Search

Search documents by keyword, phrase, regular expressions and many other parameters. Search parameters include exact word/phrase matching, case-sensitive matching, whole-word-only, begins-with/ends-with, and wildcard support. Matches are highlighted for easy viewing, and persistent hit highlighting allows you to see multiple search terms within a document.

## Customization

PrizmDoc's extensive API gives your application full control over almost every feature. You can easily personalize PrizmDoc to meet your application's needs and brand standards.

## Pre-Convert Your Documents for Even Faster Viewing

By pre-converting your important documents, your customers can experience nearly instantaneous viewing.

## Add-on Licensed Options for Self-Hosted

In PrizmDoc v12.0, we introduced feature licensing options providing customization and cost optimization so that you can decide which features are important for your customers. Please note that this section applies to PrizmDoc Self-Hosted installations only. PrizmDoc Accusoft Cloud-Hosted already includes all of these features.

## Microsoft Office Conversion

Added in PrizmDoc v12.0, the Microsoft Office Conversion service allows customers to have native rendering of Microsoft Office documents in PrizmDoc's Viewer, including Word, Excel, and PowerPoint. This enhanced rendering offers a solution for companies that require native rendering of Microsoft Office documents. What does this mean exactly? Near-native rendering produces an interpretation of the original Microsoft Office document. Background screens, font spacing, and charts may not be identical to how the document looks when viewed in the original authoring application. Native rendering presents the document as viewed in the original authoring application. This provides a true, accurate, and intended depiction of the original file.

Additional steps required to take advantage of this new service:

- Purchase the add-on license for Microsoft Office Conversion.
- Install a Microsoft Office license on each PrizmDoc server that has the Microsoft Office Conversion service enabled.

When a Microsoft Office document is sent to PrizmDoc to be processed, it will then use Microsoft Office for native rendering of that document.

## Form Field Detector

Added in PrizmDoc v12.0, the Form Field Detector recognizes a form (PDF AcroForm or raster file) then automatically creates interactive fill-able form fields.

- **PDF AcroForm:** When a PDF containing AcroForm fields is uploaded into the form converter, the form converter will automatically detect that the document contains AcroForm fields, and the form converter will not only automatically recognize the form fields, but will also identify the type of information to be collected in that form field (e.g., text input field, check box, date, initial, and signature).
- **Raster File:** When a TIFF or BMP document is uploaded, the Form Field Detector will recognize a box, underscore line, and/or a low dash line and create an interactive form field. If the Form Field Detector cannot detect a particular form field automatically, the user can easily add the field manually.

The Form Field Detector saves users precious time and money by converting paper and PDF forms into digital, interactive online documents. Conserve paper, save time, and avoid the hassle. With the Form Field Detector, you can build a document management system that works for you! Process forms and manage your documents digitally. The process becomes much more organized when you don't have to print the form, manually fill it out, sign the document, and return it completed via fax or scanner. Forms may then be built into a document management solution and sent out for filling and/or signature. You can then embed the forms into a website or application, and build in functionality to route entered data to a database for an automated upload of information.

## About PrizmDoc Help

As of PrizmDoc v12.0, we're providing the online help on our website only (rather than including an offline version in the PrizmDoc installer). This allows us to provide you with more frequent updates and ensure that you always have the latest version. In addition, the [PrizmDoc Documentation](#) page on our website provides:

- A downloadable PDF if you need an offline or printable version of the help.
- Previous versions of the help.

## What's New?

This section contains information on what's new for each release:

- [Release v12.3](#)
- [Release v12.2](#)
- [Release v12.1](#)
- [Release v12.0](#)

## Release v12.3

PrizmDoc v12.3 introduces the following new features:

## E-Signature Module

- You can limit the number of characters entered into a Form field.
- You can change the role name or color of Form fields.
- You can easily modify Field types in the Form Designer.
- You can align Fields horizontally, left, center, or right.
- When you type in a multi-line field, the font will auto-size to fit the text within the defined box.

## PrizmDoc Server

- Viewing Packages that are created with **pre-conversion** now support Server-Side Search for increased performance.
- **Server-Side Search** is now turned on by default. This should result in significantly improved performance when searching documents greater than 80 pages.
- Microsoft Office Conversion connectivity to PrizmDoc Servers is now supported for running on Linux. While the Microsoft Office Conversion add-on requires PrizmDoc Server running on Windows, it is now possible to **configure PrizmDoc Servers running on Linux** to use the Microsoft Office Conversion service. This ability allows native rendering of Microsoft documents in PrizmDoc's Viewer.
- The Content Conversion Service now provides an API to apply a unique, dynamic text stamp (based on page numbering for each page in a header and footer). This feature could be used as a technique for unique **Bates Stamping** across multiple documents.

## New **Get Started with PrizmDoc Guide**

The documentation has been updated with a helpful three-step guide to get PrizmDoc installed, configured, and running. You will also be able to integrate a basic viewer into your web app.

## Release v12.2

PrizmDoc v12.2 introduces the following new features:

### Viewer

- You can now programmatically set the **initial zoom level** of the Viewer before any content is displayed.
- The `addMarkFromSearchResult` API has been added to **ViewerControl** that allows adding a text-based mark from a search result. This allows for a quick conversion of a search result into a mark.
- We have taken steps to make it possible to support Content Security Policy (CSP) compliance in the Viewer. Please contact **Customer Support** for additional details.
- E-Signature Improvements
  - We've added support for adding **input masks** to a text box. You can require a signer to fill in the text box with specific information, such as a Phone Number, Social Security Number, Date, Zip Code, or Model Number.
  - Fields can now be marked as **"Read Only"** in the Template Designer. If a field is marked as "Read Only", the content cannot be modified in the Signing Module. This is useful in scenarios where data is filled in programmatically and the developer doesn't want the end user to modify the field.
  - The `selectEditorText` API has been added to **ViewerControl** to allow selecting the editable text within a mark's text editor.

- We've added APIs that allow setting/getting **filled-in field data** in the Signing Module. This makes it possible to set these values easily for your particular scenarios.

## PrizmDoc Server

- You can now programmatically cancel long-running search tasks (such as searching on very large documents) for [PrizmDoc Application Services](#) and [PrizmDoc Server](#).
- The [Content Conversion Service](#) now supports **conversion of password protected** PDF and Office documents.
- Two new central configuration parameters are now available to define the background color for viewing CAD documents:
  - [fidelity.vectorBackgroundColor.view.default](#) - can be used for those CAD cases when the background is not specified in the document (most CAD files except DGN).
  - [fidelity.vectorBackgroundColor.view.override](#) - can be used when the intention is to ignore the background color specified in the document.
- The Content Conversion Service now reports more granular "percentComplete" status when converting documents (previously it was only reporting 0% and 100%).

## Release v12.1

PrizmDoc v12.1 introduces the following new feature:

### CentOS 7 Support

To support the needs of our customers, we are pleased to announce we are now supporting CentOS 7 with the release of PrizmDoc 12.1.

## Release v12.0

PrizmDoc v12.0 introduces the following new features:

### Form Field Detector

The E-Signature Viewer now comes with the Form Field Detector, which is able to automatically detect fields in both PDF and raster documents.

To support the Form Field Detector feature on the server, we have added new APIs to our extensive collection of APIs (there's a [Form Extractors API for PAS](#), and a [Form Extractors API for PrizmDoc Server](#)). This new set of APIs provides developers with the capability to detect various types of form fields and automatically convert those into PrizmDoc form fields. The product supports processing both PDF Interactive Fields (commonly referred to as AcroForms) and raster documents.

See [How to use the Form Field Detector](#) for more information.

### Large Document Viewing

The Viewer has been modified to manage page memory much more judiciously, making it possible to

view very large documents (5000+ pages).

## Server-Side Search

The Viewer can now be configured to use the new Server-Side Search functionality, thus enabling text searches in very large documents. The Viewer can accommodate large search result sets by paginating and limiting the number of search results displayed in the search pane at one time. This configurable parameter ([searchResultsPageLength](#)) is set to display 250 search results on a page by default.

Server-Side Search has also been added to the services, making it possible to do text searches across very large documents (up to thousands of pages). This required the addition of new services to manage this process, and the addition of MongoDB to our installation in order to handle the text indexing we're performing. (Note that this option is configured as OFF by default in v12.0; you must specifically configure your code in order to turn it on.) It is also important to note that in an upcoming release, the Server-Side Search feature will be ON by default, and we will remove the option to choose between search modes. We want to give customers the option to experiment, test, and deploy Server-Side Search and provide feedback before making this a permanent change to the product.

See [Working Effectively with Large Documents](#) for details.

## Microsoft Office Conversion

Our new Microsoft Office Conversion is an add-on option for PrizmDoc v12.0 Server running on Windows. This new 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 that require native rendering of Microsoft Office documents. Initially, this option will be available for PrizmDoc Server running on Windows. MSO Conversion connectivity to PrizmDoc Server running on Linux will be available in a later release.

See [Natively Render Microsoft Office Documents](#) for more information.

## Viewing Session Lifetime Additions

In PrizmDoc v12.0 there are two new additions to the [Viewing Sessions API](#) that control the lifetime of viewing sessions.

- The new POST ViewingSession minSecondsAvailable property allows the user to define the lifetime of a viewing session so that it exists beyond the lifetime defined by the default configuration (20 minutes).
- The new DELETE ViewingSession API allows the user to remove a viewing session prior to its scheduled expiration time.

## Online Help

The [PrizmDoc online help](#) on our website is now continuously updated with the latest content. If you need to view older versions of the PrizmDoc help files, you will be able to access them from the same location.

## Overview of PrizmDoc

The PrizmDoc help is organized as follows to help you get going with your solution as quickly as possible:

**Understand the Core Components of PrizmDoc** - This section describes all components of PrizmDoc at a high level in order to give you a basic understanding of how the system works. If this is your first time using PrizmDoc, it is the recommended place to start.

**Server Hosting Options** - Before you install PrizmDoc, read this section to understand the different installation options. This section will also walk you through the installation of the product for the option that you choose.

**Getting Started with PrizmDoc** - Once you have installed PrizmDoc, this section will help you get the product samples running and show you how to integrate the PrizmDoc Viewer with your web tier.

**Viewer Guide** - These user guides explain how to use the PrizmDoc Viewer and are directed to end users with no programming experience. However, even if you are a developer, it is worth reading to understand the capabilities of PrizmDoc.

**Administrator Guide** - This section will help you get the most out of PrizmDoc. It covers several administrative topics including product configuration, licensing, and troubleshooting.

**Developer Guide** and **API Reference** - If you want to customize PrizmDoc or use any of the client or server APIs, these sections have numerous examples and references to everything you will need to unleash the full capabilities of the product.

## 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.

### 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	*.xlsx, *.xlsm, *.xltx, *.xltm	Yes	Yes

Format	File Extension	Supported by Microsoft Office renderer	Auto Detection
format			
Microsoft PowerPoint format	*.ppt, *.pot, *.pps	Yes	Yes
Microsoft PowerPoint Open XML format	*.pptx, *.pptm, *.potx, *.potm, *.ppsx, *.ppsm	Yes	Yes
Microsoft Visio Drawing format	*.vsd	No	Yes
Microsoft Visio XML Drawing format	*.vsdx, *.vsdm, *.vdx	No	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

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

# PrizmDoc v12.3 - Updated June 23, 2017 10

## 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.

## 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. Autodetection 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

# PrizmDoc v12.3 - Updated June 23, 2017 11

Format	Compression	File Extension	Auto Detection
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 (Windows only)	Uncompressed, RLE	*.wmf	Yes
Enhanced Metafile format (Windows only)	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
Scitex Color Tone format	Uncompressed	*.sct	Yes
WordPerfect Graphics Metafile format	RLE	*.wpg	Yes

# PrizmDoc v12.3 - Updated June 23, 2017 12

Format	Compression	File Extension	Auto Detection
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	RLE	*.cut	No

Currently Windows Metafile and Enhanced Metafile formats are only supported by PrizmDoc Server running on Windows platforms. These formats are not supported on Linux platforms. Dr. Halo (CUT) format cannot be reliably auto-detected due to its poor file signature, therefore it is identified by file extension. Currently 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

 PrizmDoc uses webfonts. Do not disable webfonts in your browser.

#### iOS 10

- Safari
- Google Chrome

#### Android 5, 6, 7

- Android Browser
- Google Chrome
- Mozilla Firefox
- Dolphin

#### Windows

- Internet Explorer 11
- Microsoft Edge
- Google Chrome
- Mozilla Firefox

#### Mac OS

- Safari

# PrizmDoc v12.3 - Updated June 23, 2017 13

- Google Chrome
- Mozilla Firefox

## Third-Party Dependencies

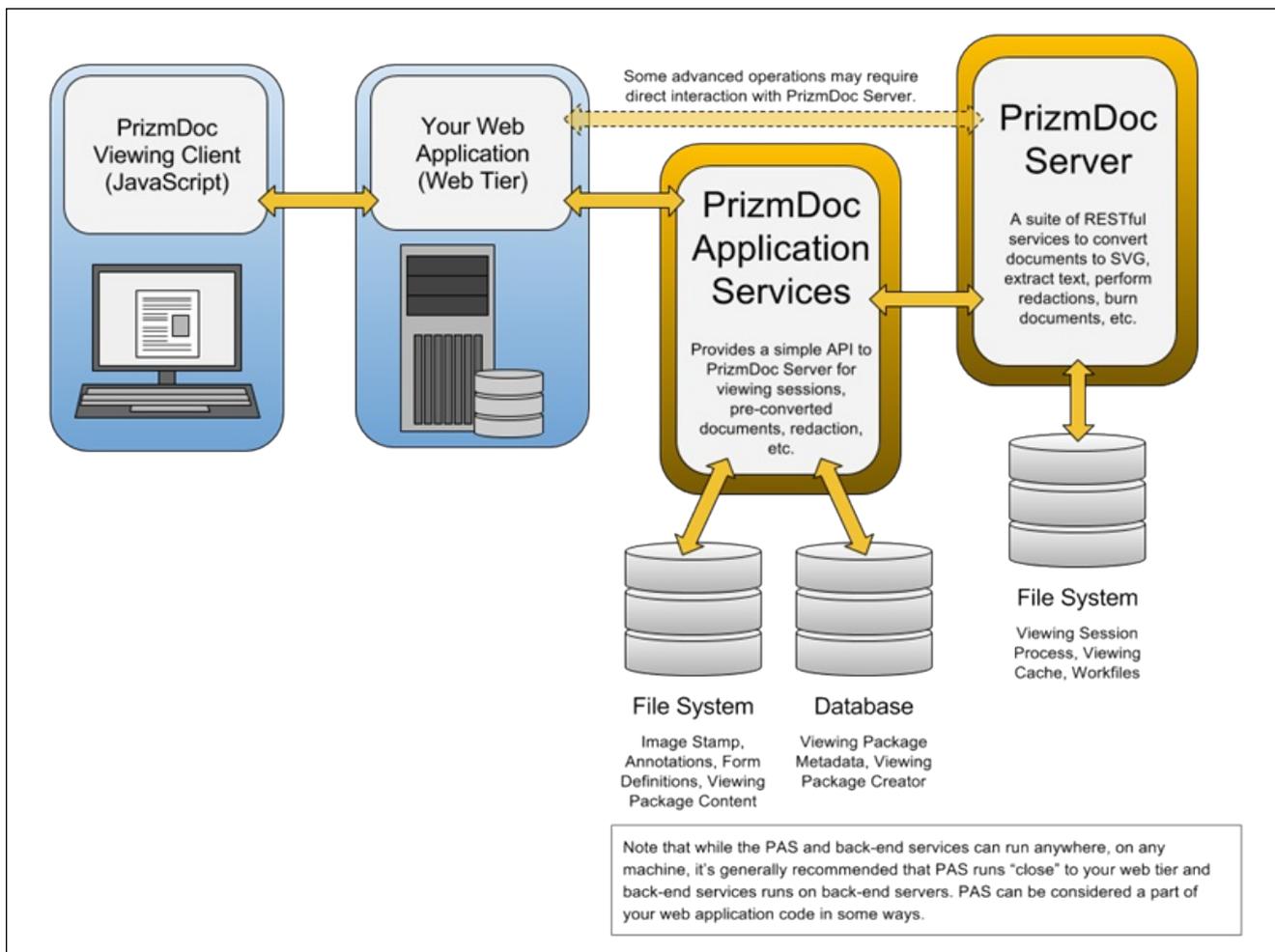
- jQuery 1.7.2
- Underscore 1.6.0
- (Full Viewer) jQuery.Hotkeys

## Understand the Core Components of PrizmDoc

PrizmDoc is a powerful, scalable platform for converting, viewing, searching, annotating and, redacting documents in hundreds of formats in a zero footprint Viewer. At its core, the basic concept of PrizmDoc is fairly straightforward: your web application passes a document to an http service which converts it into SVG and returns it to the browser. So long as the browser supports SVG (which all modern browsers now do), the document is viewable without needing to install any software on the browsing device. This model works well across PCs, tablets, and even smart phones; every device can view all standard document types without installing any extra software.

This section provides an overview of the core components of PrizmDoc shown in the diagram below:

# PrizmDoc v12.3 - Updated June 23, 2017 14



For more information, refer to the topic, [Configuring PrizmDoc Application Services in your Server's Entry Point](#).

## Viewer

PrizmDoc ships out of box with a highly evolved Viewer for displaying, searching, annotating, redacting, and commenting on your documents. Based on responsive design patterns, you can use this single viewer for desktop and mobile devices, eliminating the need to write costly additional code to support various browsers and platforms.

We have designed the Viewer to be supported both as a fully out of box product, requiring little or no customization by the developer/integrator to get your application up and running quickly, as well as a completely customizable interface with open markup and a rich API which allows you to use as much or little as our out of box viewer as you like. By maintaining maximum flexibility we are sure that our solution will meet your needs, no matter how basic or complex they may be.

There are 4 ways you can interact with the Viewer:

- **Just add it and Go!** - 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.
- **Customize using configuration parameters** - The Viewer has a number of configuration options

# PrizmDoc v12.3 - Updated June 23, 2017 15

which 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.

- **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.
- **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.

## Viewer Basics

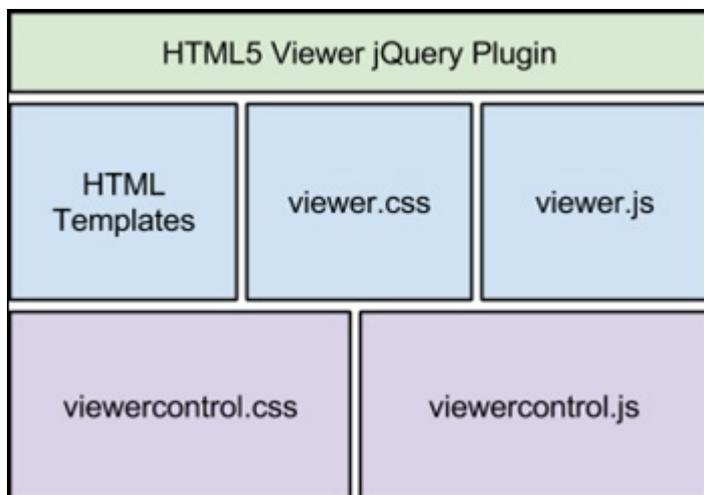
### The jQuery Plugin

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 jQuery plugin is built 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 following diagram shows the different layers:



- **HTML Templates** - the Viewer UI is designed as a group of HTML templates, which allow you to completely customize how the Viewer looks and behaves by modifying the HTML Markup.
- **viewer.css** - Viewer styling is also completely customizable using css. Customers may want the Viewer color and styling to blend seamlessly with their containing application. Allowing customization of css makes this possible.
- **viewer.js** - We expose unobfuscated Javascript code to allow customization of viewer behavior. The viewer.js is built on top of our viewer API. For complete customization, you could build your own viewer using our API if you desire.
- **ViewerControl.js** - This is the core component to any viewer. It implements the logic of document display, mouse tools and touch interaction, search, printing, annotations, and redactions. It is

# PrizmDoc v12.3 - Updated June 23, 2017 16

responsible for calling to the PrizmDoc Server via PrizmDoc Application Services, to retrieve document and annotation data. It renders a UI - the page list - which permits scrolling through the content of a document. This code is not intended for customization, as everything you need to do should be available in viewer.js.

## Web Tier

The Web Tier, for purposes of this discussion, is your web application. You will embed the Viewer and its required resources within your web application. PrizmDoc ships with integration code in several languages to help you get started quickly.

Integration with the PrizmDoc Application Services (PAS) RESTful API is straightforward and provides a clear and concise interface for your document viewing needs. Additionally, samples illustrate the use of a fully functional Viewer inside of a sample project in your preferred language that contains the minimal code necessary to connect to the PrizmDoc Application Services (PAS) RESTful API. For each of the supported language integrations, we provide integration topics in the section: [Integrating the Viewer with your Application](#).

## PrizmDoc Application Services (PAS)

### PrizmDoc Application Services (PAS) Overview

PrizmDoc Application Services (PAS) is a service that provides application-level logic for the Viewer; such as enabling document viewing through the PrizmDoc Server, saving and loading of markup, and handling opening of documents and creating viewing sessions. This service is designed to be a gateway between the Viewer and PrizmDoc Server, and takes the place of the Web Tier samples available in previous versions of the product.

The PrizmDoc Application Services are intended to run on the same server as your web tier, although that is not required. As it is a standalone service, it can run anywhere between your web application and PrizmDoc Server, including on its own separate machine if necessary.

### Architecture in the Out-of-Box Product using PrizmDoc Application Services

The only requirement that PAS has is that a single, configurable endpoint from the Viewer is proxied to PAS. If logic beyond the default behavior is required (such as customized storage of markup, for example), PAS exposes a flexible API allowing the user to handle only very specific viewer requests while delegating all others to PAS.

With the introduction of PrizmDoc Application Services, the Web Tier Samples in PrizmDoc only include a small amount of proxy code, which accomplishes the forwarding of viewer requests to PrizmDoc Application Services. This provides minimal code for customers to implement, and it is code that should not need updates often.

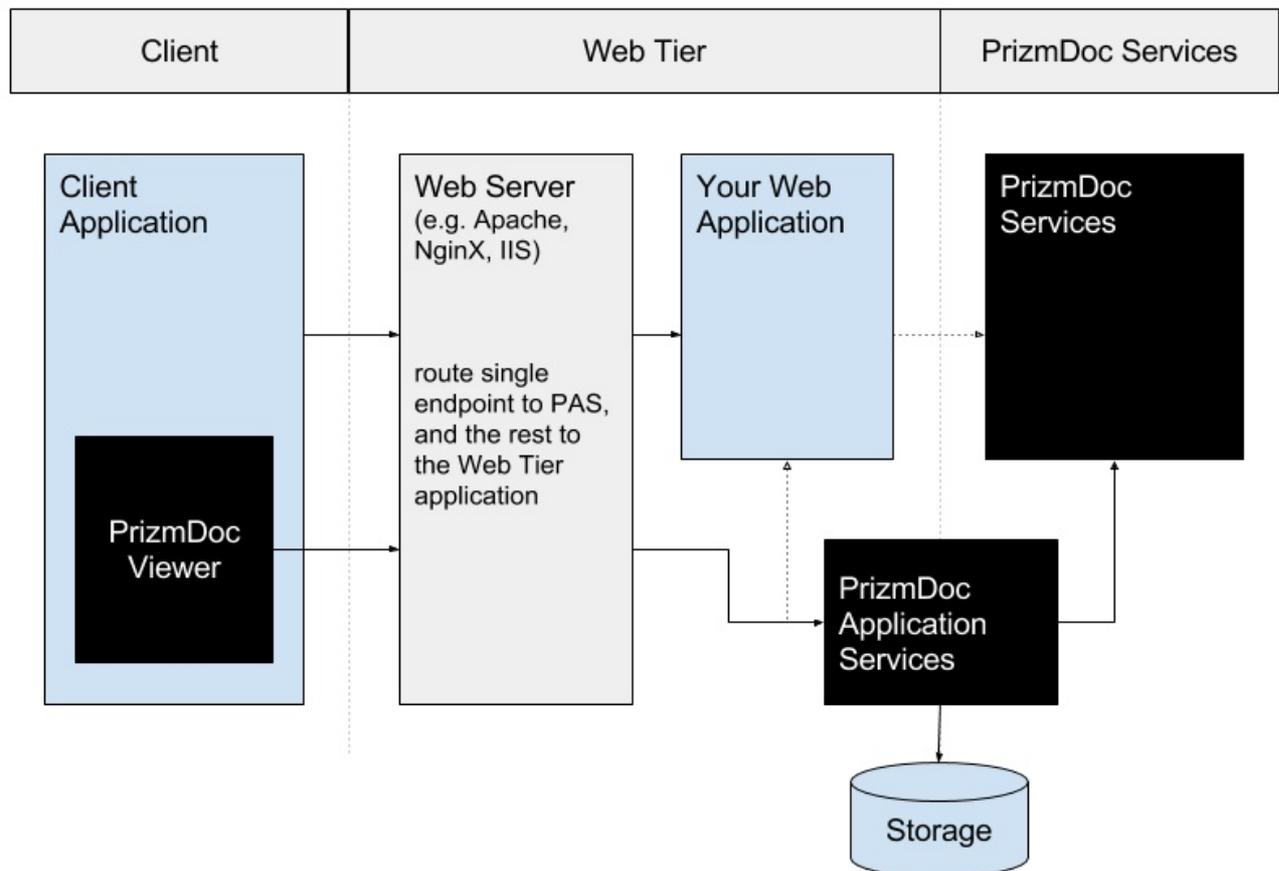
### Web Server of Your Choice

The minimal code that comes with the Web Tier Samples is only an example; any proxy into PAS can accomplish

# PrizmDoc v12.3 - Updated June 23, 2017 17

the same task with the same minimal code. For a more performant solution, we would encourage you to handle proxying through your web server of choice (such as Apache, NginX, or IIS), allowing your web application to only contain your own logic, without the need to integrate any Accusoft sample code.

This is outlined by the architecture below:



For more information on how to configure the PrizmDoc Application Services, refer to the topic [Administering PrizmDoc > PrizmDoc Application Services](#).

## Pre-Conversion and Caching with Application Services

In PrizmDoc v11.0 we introduced new features which support the ability to do pre-conversion and long term caching of documents in a central location. To make use of this new functionality, an external instance of a database will need to be configured to run with PAS. The introduction of PrizmDoc Application Services handles session management and other stateful operations and lets the PrizmDoc Server handle the document conversion and processing.

## Using PAS in a Multi-Server Environment

PAS is designed to scale out well. In order to install and run PAS on multiple servers, you will need to consider the following:

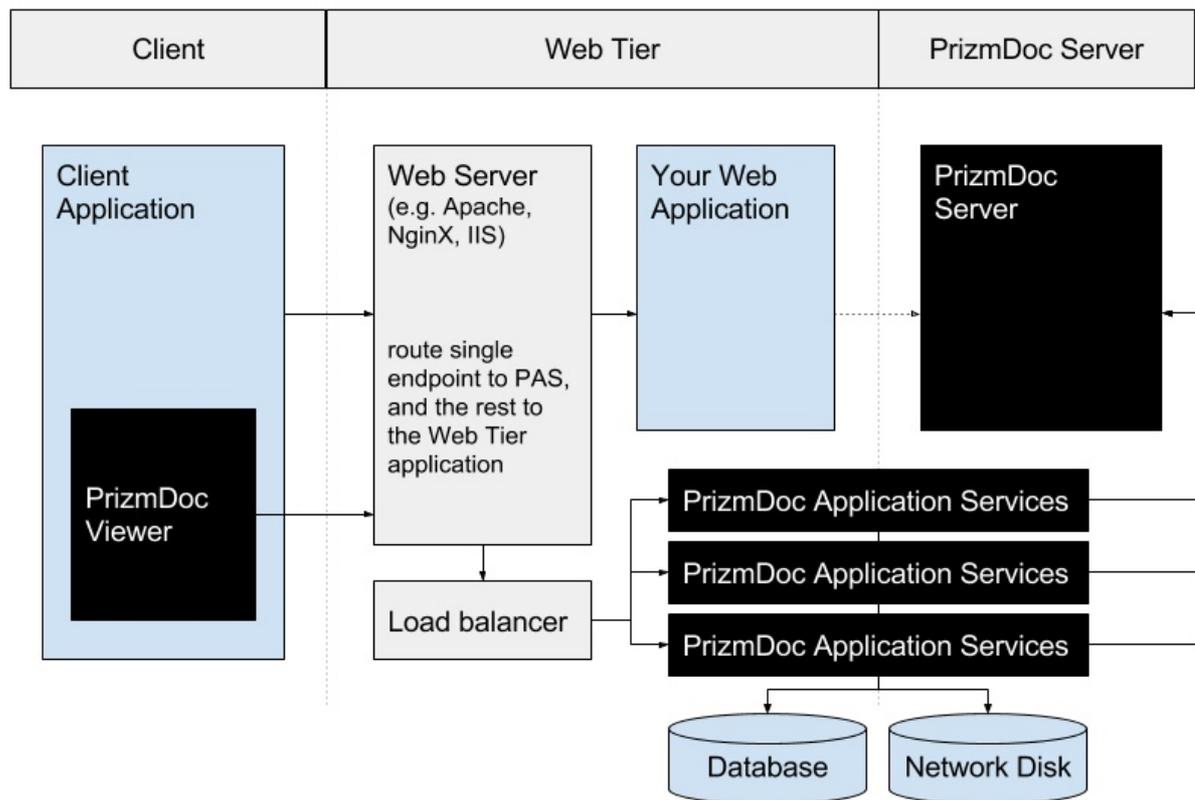
- You will need a load balancer to accept incoming requests and route them to any instance of PAS. Since any request can be handled by any instance in the cluster, any on-the-shelf load balancing solution should do the job.
- All instances of PAS will need to be configured with the same PrizmDoc Server installation (for single-server

# PrizmDoc v12.3 - Updated June 23, 2017 18

or multi-server).

- 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](#).

The following is an illustration of this architecture:



For more information on installing PAS on a multi-server cluster, refer to the topic [Run PrizmDoc Application Services on Multiple Servers](#).

## PrizmDoc Server (Self-Hosted)

With the introduction of PrizmDoc 11, the PrizmDoc Server can be installed by you on-premise or hosted by Accusoft. For more information on deployment options, read the topic [Server Hosting Options](#). By using the Accusoft Cloud-Hosted solution, you don't need to know anything about the components of the PrizmDoc Server. This topic is geared to those who will be installing the Back-end on-premise:

- The PrizmDoc Server is a collection of services that perform several important functions such as: Conversion of documents from the source file to SVG or raster, depending on the source document content, for display in the Viewer. The conversion of documents into a common format is the key to viewing many different file types in a single viewer.
- Content Conversion Services RESTful API. Allows conversion of content from all supported formats into PDF or TIFF output. Content Conversion Services also allows you to combine documents (or pages within documents) from disparate formats into a new document. Through Content Conversion

# PrizmDoc v12.3 - Updated June 23, 2017 19

Services, you can also add custom headers and footers to your document, all through a RESTful API.

- Redaction burning with the MarkupBurner API.
- Reporting on system errors with the Error Reporting Service.
- Reporting on system health which is managed by the Watchdog Service.

## PrizmDoc Server (Accusoft Cloud-Hosted)

In PrizmDoc v11.0, we introduced the ability to bypass installing and maintaining the PrizmDoc Server and take the advantage of using our Accusoft Cloud-Hosted Back-end. This allows you to leave the server management to Accusoft, ensuring that you can focus more of your resources on developing your application instead of administering servers. More information can be found under the [Server Hosting Options](#) topic, but in short, you may want to consider this option if you:

- Need to quickly get up and running with a viewer prototype without committing to server procurement during your development cycle.
- Have a cloud-based product where you would prefer an Accusoft Cloud-Hosted solution.
- Have a mix of hosted and cloud applications/deployments and want to leverage your development investment to handle both scenarios with one front end.

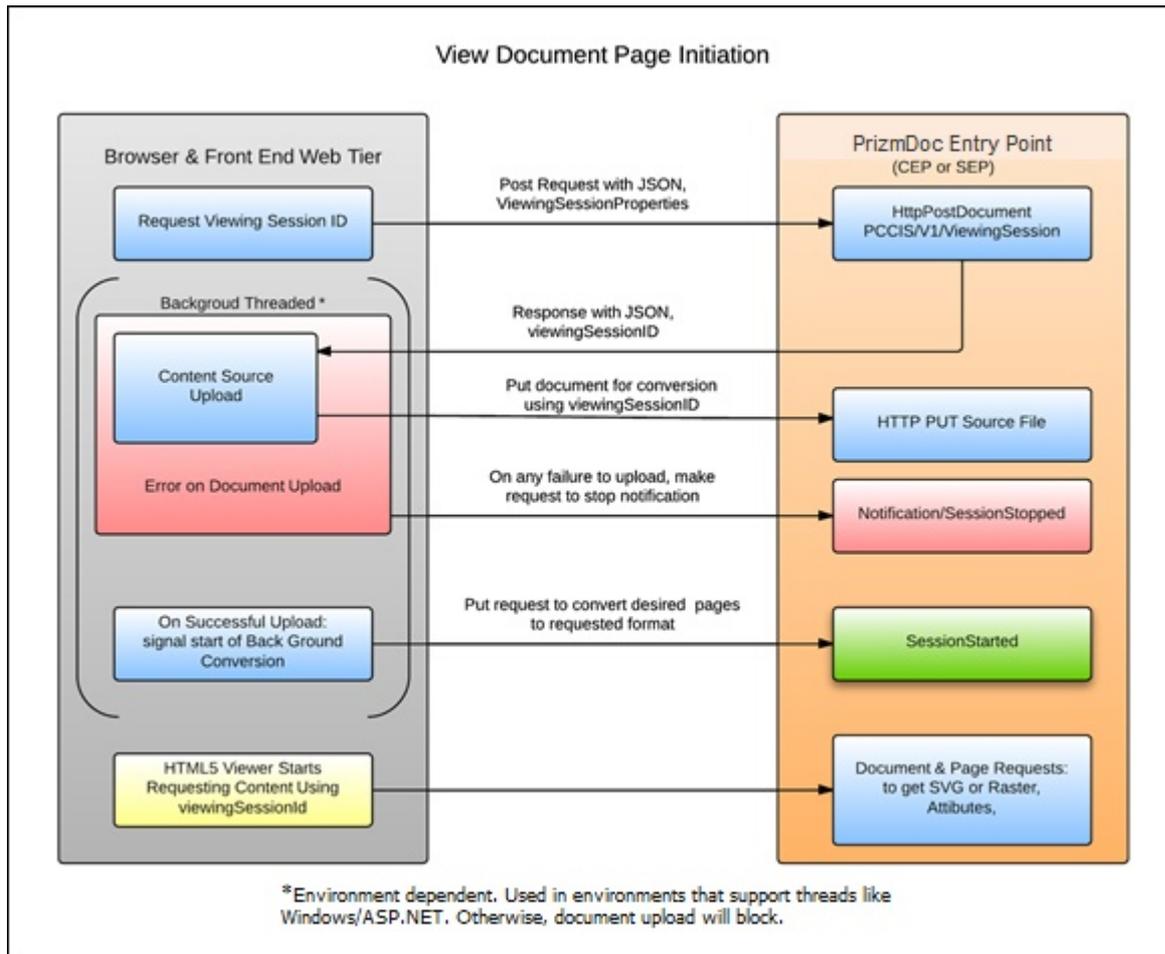
With PrizmDoc, you can write all of your viewer and web tier code without having to worry about deployment scenarios. Moving back and forth between Accusoft Cloud-Hosted or Self-Hosted backends is seamless. Of course, separate licensing may be required.

## Document Workflow

The PrizmDoc Server architecture is based on a two-tier web solution where the SEP, or CEP, is the backend tier and the front end tier is a light-weight-pass-through tier keeping the customer's web application simpler to implement the PrizmDoc Server. There is a slight cost to this approach in that source documents need to be uploaded from the front end web tier to the PrizmDoc Server tier. The recommendation is to embed code into the web page to initiate the upload sequence of the document and start the PrizmDoc Server preparing the document for viewing. Another caveat to consider is that the document identification needs to be supplied by PrizmDoc Server so that it can better manage that resource once it is uploaded.

The recommended process for document viewing:

# PrizmDoc v12.3 - Updated June 23, 2017 20

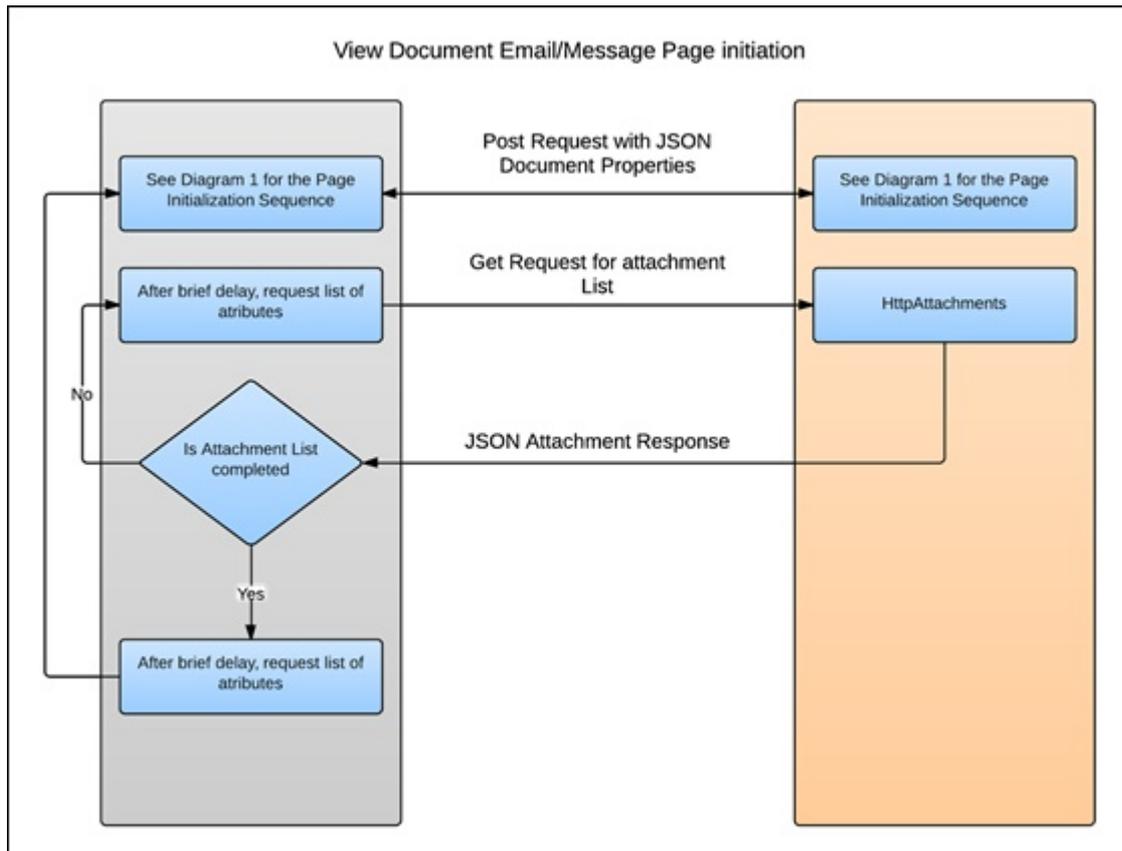


1. Post a **JSON String object** of type **DocumentProperties**.
2. Use returned JSON string **viewingSessionId** value as the document identifier for viewing session.
3. Open **document** and write its content into a stream and upload.
4. Put request to **convert document** into viewable web pages.
5. Setup **Viewer control** on Web page with the **document identifier** returned from the Post above.

The viewing of email or message files is a more complicated process where the body of those types must be dissected into individual pieces for any attachments. Each attachment has its own document ID (now called **Viewing Session Id**) which must be handled separately. The web page must determine if a **document** query string parameter or a session identifier query string, **viewingSessionId**, is being used as the query string. In general, the **viewingSessionId** is the identification element for correspondence with PrizmDoc Server.

The general concept is illustrated below:

# PrizmDoc v12.3 - Updated June 23, 2017 21



## Next Steps

Now that you have been given an overview of PrizmDoc, what's next?

- Read the section on [Server Hosting Options](#) to determine which installation is right for you.
- Read the [Getting Started](#) topics to install the product and run the Viewer samples. We also show you how to quickly get the Viewer embedded in your web application.
- The [Viewing End User Guides](#) are useful for understanding the power of the Viewer out of the box.
- From there, you can dive into [Developer Guides](#) if you are customizing PrizmDoc.

## Server Hosting Options

### We Host It or You Host It!

With PrizmDoc, it has never been easier to get up and running by taking advantage of our Accusoft Cloud-Hosted Services. We now offer two options for hosting your PrizmDoc Server. Before you move forward with installation, you will want to understand these options and determine what is right for you.

#### Self-Hosted (On Premise)

The self-hosted installation is the traditional hosting option, in which you install PrizmDoc Server entirely

# PrizmDoc v12.3 - Updated June 23, 2017 22

on your own infrastructure. PrizmDoc Server is flexible and scalable in that it can be installed on single server or a multi-server scenario, with several different possible options for multi-server installs.

## Accusoft Cloud-Hosted

We now offer the option to use the Accusoft Cloud-Hosted Services, in which you only need to install a lightweight web tier and Viewer to connect to our PrizmDoc Server, hosted on AWS. This allows you to have all the power of our customizable Viewer and set of APIs, without having to provision the hardware necessary to run PrizmDoc Server on-premise.

## Reasons to Choose Accusoft Cloud-Hosted

- **Rapid integration** - Using the Accusoft Cloud-Hosted Services, you can focus on getting the PrizmDoc Viewer integrated within your application quickly without having to install PrizmDoc Server. The same Viewer and APIs work interchangeably with the Self-Hosted or Accusoft Cloud-Hosted Services. This means that you can focus more on adding value to your application and less on administration.
- **Evaluation and prototyping** - The Accusoft Cloud-Hosted option is a great solution if you are not yet ready to provision server hardware while you complete your evaluation or development cycle. This allows you to plan for your server architecture, while you develop, so you are ready to launch when you deploy on-premise with a small configuration change.
- **You have a cloud-hosted solution** - Accusoft has experience with hosting PrizmDoc in the cloud and we are ready to support your cloud-based application. PrizmDoc functionality is fully supported in the cloud using our Accusoft Cloud-Hosted Services.
- **You are unsure whether you will need to host it on-premise or in the cloud** - With a common set of APIs for Cloud-Hosted or Self-Hosted, you don't have to decide now. You can change your hosting strategy anytime without having to re-develop your application.

## Reasons to Choose Self-Hosted

- **You need your data on your own infrastructure and network.** In this case, you may still want to evaluate the product with the Cloud-Hosted Services but move to your own server architecture for production.
- **You need more granular control over server configuration options.** You may want to take advantage of advanced server configuration or server-side APIs. In this case, the self-hosted option may be better for you.

## What happens if I need to change my decision later?

As mentioned, both the Cloud-Hosted and Self-Hosted options are based on a common architecture. Moving from one option to another requires a minor configuration entry and an appropriate license.

## Installers Shipped with PrizmDoc

Beginning with PrizmDoc v11, the Windows installers have changed. Previously, the Viewer and Server installers were bundled into a single executable package. Since most users install their web tier and server components on a separate server, we have split the Viewer and Server into separate installers to better fit our customer's use cases. Further, if you are using the Cloud-Hosted option, you will notice a much lighter download for the Viewer installer.

The Viewer installer contains all of the Viewer code and the Web Tier samples. During the install you will have the option to connect to the Cloud-Hosted Services. With this option, you will not need the PrizmDoc Server installer. However, if you choose to host the PrizmDoc Server on-premise, you should

# PrizmDoc v12.3 - Updated June 23, 2017 23

run the separate PrizmDoc Server installer first, and then run the Viewer install, bypassing the Cloud-Hosted options. Please refer to the [Getting Started with PrizmDoc](#) section for complete instructions.

## Licensing

The license for the Cloud-Hosted option is distinct from the Self-Hosted options. When you sign up for a trial, you will receive separate license keys to support either option. You can convert your trial Cloud-Hosted license to a paid license on our [website](#) or consult a [sales representative](#) to purchase an on-premise license.

## Contact Accusoft Support

For information about Accusoft Support, please refer to the [Software Support and Maintenance Policy](#) on the Accusoft web site or call Accusoft at 813-875-7575.

Call the Accusoft Sales department at 1-800-875-7009 (press 1) to address any questions concerning PrizmDoc pricing, licensing, or options. Contact information for international customers can be found here: [Customer Support Contact Form](#).

## Glossary

This section contains the following information:

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

## Definitions

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

Term	Also Known as...	Definition
Cloud-Hosted	-	A PrizmDoc Server hosted by Accusoft. See <a href="#">PrizmDoc Server (Accusoft Cloud-Hosted)</a> .
Application Services	-	The component of PrizmDoc that provides application-level logic between PrizmDoc Server and the Viewer or the customer's web-tier. See <a href="#">PrizmDoc Application Services</a> .

# PrizmDoc v12.3 - Updated June 23, 2017 24

AutoRedaction Service	ARS	A component of PrizmDoc Server that creates markup XML for redactions on a document. See <a href="#">Performing Auto-Redaction</a> .
Central Configuration	Central Config	The configuration file for PrizmDoc Server. See <a href="#">Central Configuration</a> .
Cloud Entry Point	CEP	The Load Balancer endpoint for requests to a cluster of PrizmDoc Server instances, to be routed to an appropriate instance.
Content Conversion Service	CCS	A component of PrizmDoc Server that provides document format conversion. See <a href="#">Convert Content with CCS</a> .
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 <a href="#">Error Reporting</a> .
Format Detection Service	FDS	A component of PrizmDoc Server that identifies the format of a source document. See <a href="#">Format Detection Configuration &amp; Use</a> .
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 Multi-Server Mode.
Multi-Server 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 <a href="#">PrizmDoc Multi-Server Mode</a> .
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.

# PrizmDoc v12.3 - Updated June 23, 2017 25

Prizm License Utility	PLU	A component of PrizmDoc Server that handles licensing of the product on installation. See <a href="#">Licensing</a> .
PrizmDoc	-	The full product composed the PrizmDoc Server, Application Services, and the Viewer.
PrizmDoc Server	-	A back-end component of PrizmDoc that performs conversions and other 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.
Self-Hosted	-	A PrizmDoc Server hosted on a customer's server. See <a href="#">PrizmDoc Server (Self-Hosted)</a> .
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 that allows the uploading and display of documents through a browser. See <a href="#">Viewer</a> .
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.
Work File Service	WFS	A component of PrizmDoc Server that handles the uploading and storage of source documents.

## New Terms

With v11.0 and later, the product names have been updated to the following:

### Accusoft Cloud-Hosted Services

Formerly referred to as:

- Accusoft Cloud Services (ACS)

# PrizmDoc v12.3 - Updated June 23, 2017 26

- Accusoft-Hosted Services

## PrizmDoc Application Services (PAS)

Formerly referred to as:

- Prizm Application Services

## PrizmDoc Server

Formerly referred to as:

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

## The Viewer

Formerly referred to as:

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

## Legal

This section contains legal information for PrizmDoc:

- [Copyright Information](#)
- [Software License Agreement](#)

# PrizmDoc v12.3 - Updated June 23, 2017 27

- [Third-Party Attributions](#)

## Copyright Information

©1996-2017 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 USA.

Accusoft Corporation  
4001 North Riverside Drive  
Tampa, FL 33603  
Sales: 813-875-7575  
[www.accusoft.com](http://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.

Microsoft, Internet Explorer, Microsoft.NET, Silverlight, Visual Basic, Visual Studio, and Visual Studio.NET are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Itanium is a registered trademark of Intel Corporation in the United States and other countries.

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

## Software License Agreement

### ACCUSOFT CORPORATION

#### PRIZMDOC 'SHRINK-WRAP' LICENSE AGREEMENT

PLEASE READ THIS LICENSE AGREEMENT ("AGREEMENT") WHICH GOVERNS YOUR RIGHT TO USE OF PRIZMDOC ("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 CHECKING THE "I ACCEPT THE TERMS IN THE LICENSE AGREEMENT" CHECKBOX AND THEN CONTINUING WITH THE INSTALLATION BY PUSHING THE "NEXT" BUTTON, YOU AND LICENSEE AGREE TO BE BOUND BY THIS AGREEMENT.

**1. Background.** Accusoft Corporation, a Florida corporation, ("Accusoft") is the owner of all right, title, and interest in the software system known as PrizmDoc ("Program") and consisting of an installed front-end component plus either an installed back-end component hosted by LICENSEE or back-end

# PrizmDoc v12.3 - Updated June 23, 2017 28

services ("Service") hosted by Accusoft. LICENSEE desires to receive and use a copy of Program under the terms and conditions stated herein, for 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 all 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 expires after 30 days at which time Program will cease operating. If your evaluation is not complete at that time, please contact [www.accusoft.com](http://www.accusoft.com) or [sales@accusoft.com](mailto: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](http://www.accusoft.com) or through [sales@accusoft.com](mailto: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](http://www.accusoft.com) or through [sales@accusoft.com](mailto: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

# PrizmDoc v12.3 - Updated June 23, 2017 29

(b) the third party also agrees to all the terms of this Agreement as LICENSEE. When LICENSEE uses Service for Program back-end, LICENSEE's Commercial License includes a monthly transaction limit of up to 25,000 transactions for an Enterprise Essentials license, 50,000 transactions for an Enterprise license and up to 100,000 transactions for an Enterprise Elite license. When LICENSEE hosts the back-end component of Program, and LICENSEE has purchased the PrizmDoc Viewer level for Program, LICENSEE's Commercial License to use the Program is restricted by this Agreement to include only the viewing and searching capabilities of Program.

**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.

## **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

# PrizmDoc v12.3 - Updated June 23, 2017 30

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.

# PrizmDoc v12.3 - Updated June 23, 2017 31

**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

# PrizmDoc v12.3 - Updated June 23, 2017 32

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](mailto:sales@accusoft.com).

**24. Third Party Notices.** See [Third-Party Attributions](#).

Rev. 2016-06-09

## Third-Party Attributions

### Third Party Notices

- [Windows Server](#)
- [Linux Server](#)
- [Deprecated Viewers](#)
- [Current Viewer \(v9.x and higher\)](#)
- [Third Party Notices – Other](#)

### Windows Server

The following third party software may be used or distributed in the Windows Server component of the Program:

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>

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>

# PrizmDoc v12.3 - Updated June 23, 2017 33

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](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](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](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](http://commons.apache.org/logging/download_logging.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 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.html>

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>

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

Copyright (c) 2014 Jonathan Ong <[me@jongleberry.com](mailto:me@jongleberry.com)>

Copyright (c) 2014-2015 Douglas Christopher Wilson <[doug@somethingdoug.com](mailto: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/>

# PrizmDoc v12.3 - Updated June 23, 2017 34

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) 2010-2014 Caolan McMahon

Download: <https://github.com/caolan/async>

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

Consul

Mozilla Public License, version 2.0

Download: <https://github.com/hashicorp/consul>

License: <install directory>/license/Consul/LICENSE

cors

The MIT License (MIT)

Copyright (c) 2013 Troy Goode [troygoode@gmail.com](mailto:troygoode@gmail.com)

Download: <https://github.com/expressjs/cors>

License: <https://github.com/expressjs/cors/blob/master/LICENSE>

Duration.js

Copyright (c) 2013 Ilija 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](mailto:tj@vision-media.ca)>

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

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

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

License: <https://github.com/strongloop/express/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](http://www.freetype.org)). All rights reserved.

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

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

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

# PrizmDoc v12.3 - Updated June 23, 2017 35

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

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 API ([https://mvnrepository.com/artifact/javax.media/jai\\_core/1.1.3](https://mvnrepository.com/artifact/javax.media/jai_core/1.1.3))

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

Download: [http://central.maven.org/maven2/javax/media/jai\\_core/1.1.3/](http://central.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](http://download.java.net/media/jai/builds/release/1_1_3/LICENSE-jai.txt)

Java Advanced Imaging Image I/O Tools

([https://mvnrepository.com/artifact/com.sun.media/jai\\_imageio/1.1](https://mvnrepository.com/artifact/com.sun.media/jai_imageio/1.1))

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

Download: [http://central.maven.org/maven2/com/sun/media/jai\\_imageio/1.1/](http://central.maven.org/maven2/com/sun/media/jai_imageio/1.1/)

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

Java HTML Tidy - JTidy (<http://jtidy.sourceforge.net/index.html>)

Copyright (c) 1998-2000 World Wide Web Consortium (Massachusetts Institute of Technology, Institute National de Recherché en Informatique et en Automatique, Keio University). All Rights Reserved.

Download: <http://jtidy.sourceforge.net/download.html>

License: <install directory>/licenses/JTidy/license.txt

JavaMail 1.4.3 (<http://www.oracle.com/technetwork/java/javamail/index.html>)

Copyright (c) 2009 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Download: <http://www.oracle.com/technetwork/java/javamail/index-138643.html>

License: [http://download.oracle.com/otn-pub/java/licenses/javamail-1.4.3-oth-JPR\\_license\\_1.html](http://download.oracle.com/otn-pub/java/licenses/javamail-1.4.3-oth-JPR_license_1.html)

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>)

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

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

# PrizmDoc v12.3 - Updated June 23, 2017 36

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

License: BSD License (<install directory>/licenses/jbig2\_1.4/license.txt)

JRE 1.6.0.35 (<http://java.sun.com/products/archive/j2se/6u35/index.html>)

Copyright (c) 2006, 2010, Oracle and/or its affiliates. All rights reserved.

Download: <http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html#jre-6u35-oth-JPR>

License: <http://www.oracle.com/technetwork/java/javase/terms/license/index.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>)

Copyright (c) 2007 James Newton-King

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

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

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

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

JavaBeans Activation Framework (<http://www.oracle.com/technetwork/java/javase/downloads/index-135046.html>)

Download: <http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-java-plat-419418.html#jaf-1.1-fr-oth-JPR>

License: [http://download.oracle.com/otn-pub/java/licenses/jaf-1.1-mrel-spec-oth-JPR\\_license\\_1.html](http://download.oracle.com/otn-pub/java/licenses/jaf-1.1-mrel-spec-oth-JPR_license_1.html)

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

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

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).

# PrizmDoc v12.3 - Updated June 23, 2017 37

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

Please contact Accusoft support at [support@accusoft.com](mailto:support@accusoft.com) 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

lodash

The MIT License (MIT)

Copyright (c) 2012-2015 The Dojo Foundation <<http://dojofoundation.org/>>

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

License: <<http://dojofoundation.org/>>

merge-stream

The MIT License (MIT)

Copyright (c) Stephen Sugden <[me@stephensugden.com](mailto:me@stephensugden.com)> (stephensugden.com)

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

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

mime

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](mailto: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>

mv

The MIT License (MIT)

Copyright (c) 2014 Andrew Kelley

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

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

netty-buffer

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>

netty-codec

Apache License, Version 2.0

# PrizmDoc v12.3 - Updated June 23, 2017 38

Copyright (c) 2012 The Netty Project

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

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

netty-codec-http

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>

netty-common

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>

netty-handler

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>

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.html>)

NLog

Copyright (c) 2004-2011 Jaroslaw Kowalski

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/blob/master/LICENSE.txt>

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

node-consul

The MIT License (MIT)

Copyright (c) 2014 Silas Sewell

# PrizmDoc v12.3 - Updated June 23, 2017 39

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

License: <install directory>/licenses/node-consul/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-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](mailto:felix@debuggable.com)) and contributors

Download: <https://github.com/felixge/node-mysql>

License: <install directory>/licenses/node-mysql/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](mailto:tim@debuggable.com))

Felix Geisendörfer ([felix@debuggable.com](mailto:felix@debuggable.com))

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

license: MIT <https://github.com/tim-kos/node-retry/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](mailto: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

Copyright (c) Joyent, Inc. and other Node contributors

Download: <http://nodejs.org/dist/v0.10.26/>

Version: 0.10.26

License: <install directory>\licenses\Node.js\LICENSE.txt

# PrizmDoc v12.3 - Updated June 23, 2017 40

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://sourceforge.net/projects/openjpeg.mirror/files/>

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

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

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>

Poppler

Copyright (c) 2005-2014 The Poppler Developers

Copyright (c) 1996-2011 Glyph & Cog, LLC

Download: <http://poppler.freedesktop.org>

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

## **PrizmDoc Server Fonts:**

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

Copyright (c) by Font AC

Download: [http://www.ac-font.com/jp/detail\\_jb\\_007.php](http://www.ac-font.com/jp/detail_jb_007.php)

License: <http://www.ac-font.com/jp/terms.php>

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

# PrizmDoc v12.3 - Updated June 23, 2017 41

Download: <https://github.com/hapijs/qs>

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

rapidjson

Copyright (c) 2011 Milo Yip ([miloyip@gmail.com](mailto:miloyip@gmail.com))

Download: <https://github.com/miloyip/rapidjson>

Version: v0.11

License: <install directory>/licenses/rapidjson/license.txt

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)

sails-mysql

The MIT License (MIT)

Copyright (c) 2016 Mike McNeil, Balderdash & contributors

Download: <https://github.com/balderdashy/sails-mysql>

License: <install directory>/licenses/sails-mysql/README.md and <https://sails.mit-license.org/>

sails-sqlserver

The MIT License (MIT)

Download: <https://github.com/cnect/sails-sqlserver>

License: <https://github.com/cnect/sails-sqlserver>

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

StringSearch - high-performance pattern matching algorithms in Java

Copyright (c) 2003-2010 Johann Burkard

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

License: <http://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](http://bouncycastle.org/latest_releases.html)

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

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

# PrizmDoc v12.3 - Updated June 23, 2017 42

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

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

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

Touch.exe ([http://www.codeproject.com/KB/applications/touch\\_win.aspx](http://www.codeproject.com/KB/applications/touch_win.aspx))

Copyright (c) 2002 by Jörgen Sigvardsson

Download: [http://www.codeproject.com/KB/applications/touch\\_win.aspx](http://www.codeproject.com/KB/applications/touch_win.aspx)

License: [http://www.codeproject.com/KB/applications/touch\\_win.aspx](http://www.codeproject.com/KB/applications/touch_win.aspx)

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

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

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

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

tunnel-agent

Apache License Version 2.0

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

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

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>

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](http://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](mailto: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](mailto: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.

# PrizmDoc v12.3 - Updated June 23, 2017 43

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

yargs

Copyright (c) 2010 James Halliday ([mail@substack.net](mailto: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

## Linux Server

The following third party software may be used or distributed in the Linux Server component of the Program:

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>

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](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/>)

# PrizmDoc v12.3 - Updated June 23, 2017 44

Copyright (c) 2001-2008 The Apache Software Foundation

Download: [http://commons.apache.org/io/download\\_io.cgi](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](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 2003-2007 The Apache Software Foundation

Download: [http://commons.apache.org/logging/download\\_logging.cgi](http://commons.apache.org/logging/download_logging.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 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.html>

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>

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

Copyright (c) 2014 Jonathan Ong <[me@jongleberry.com](mailto:me@jongleberry.com)>

Copyright (c) 2014-2015 Douglas Christopher Wilson <[doug@somethingdoug.com](mailto: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) 2010-2014 Caolan McMahon

Download: <https://github.com/caolan/async>

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

Consul

Mozilla Public License, version 2.0

# PrizmDoc v12.3 - Updated June 23, 2017 45

Download: <https://github.com/hashicorp/consul>

License: <install directory>/license/Consul/LICENSE

cors

The MIT License (MIT)

Copyright (c) 2013 Troy Goode [troygoode@gmail.com](mailto: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>

expat (<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](mailto:tj@vision-media.ca)>

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

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

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

License: <https://github.com/strongloop/express/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 © 2012 The FreeType Project ([www.freetype.org](http://www.freetype.org)). All rights reserved.

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

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

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

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

Java Advanced Imaging API ([https://mvnrepository.com/artifact/javax.media/jai\\_core/1.1.3](https://mvnrepository.com/artifact/javax.media/jai_core/1.1.3))

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

# PrizmDoc v12.3 - Updated June 23, 2017 46

rights reserved.

Download: [http://central.maven.org/maven2/javax/media/jai\\_core/1.1.3/](http://central.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](http://download.java.net/media/jai/builds/release/1_1_3/LICENSE-jai.txt)

Java Advanced Imaging Image I/O Tools

([https://mvnrepository.com/artifact/com.sun.media/jai\\_imageio/1.1](https://mvnrepository.com/artifact/com.sun.media/jai_imageio/1.1))

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

Download: [http://central.maven.org/maven2/com/sun/media/jai\\_imageio/1.1/](http://central.maven.org/maven2/com/sun/media/jai_imageio/1.1/)

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

Java HTML Tidy - JTidy (<http://jtidy.sourceforge.net/index.html>)

Copyright (c) 1998-2000 World Wide Web Consortium (Massachusetts Institute of Technology, Institute National de Recherché en Informatique et en Automatique, Keio University). All Rights Reserved.

Download: <http://jtidy.sourceforge.net/download.html>

License: <install directory>/licenses/JTidy/license.txt

JavaMail 1.4.3 (<http://www.oracle.com/technetwork/java/javamail/index.html>)

Copyright (c) 2009 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Download: <http://www.oracle.com/technetwork/java/javamail/index-138643.html>

License: [http://download.oracle.com/otn-pub/java/licenses/javamail-1.4.3-oth-JPR\\_license\\_1.html](http://download.oracle.com/otn-pub/java/licenses/javamail-1.4.3-oth-JPR_license_1.html)

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>)

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

Copyright (c) 1997-2008, IDRolutions and Contributors.

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

License: BSD License (<install directory>/licenses/jbig2\_1.4/license.txt)

JRE 1.6.0.35 (<http://java.sun.com/products/archive/j2se/6u35/index.html>)

Copyright (c) 2006, 2010, Oracle and/or its affiliates. All rights reserved.

Download: <http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html#jre-6u35-oth-JPR>

License: <http://www.oracle.com/technetwork/java/javase/terms/license/index.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>

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

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

JavaBeans Activation Framework (<http://www.oracle.com/technetwork/java/javase/downloads/index-135046.html>)

Download: <http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-java-plat-419418.html#jaf-1.1-fr-oth-JPR>

License: [http://download.oracle.com/otn-pub/java/licenses/jaf-1.1-mrel-spec-oth-JPR\\_license\\_1.html](http://download.oracle.com/otn-pub/java/licenses/jaf-1.1-mrel-spec-oth-JPR_license_1.html)

# PrizmDoc v12.3 - Updated June 23, 2017 47

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

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

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 at [support@accusoft.com](mailto:support@accusoft.com) 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

lodash

The MIT License (MIT)

Copyright (c) 2012-2015 The Dojo Foundation <<http://dojofoundation.org/>>

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

License: <<http://dojofoundation.org/>>

merge-stream

The MIT License (MIT)

Copyright (c) Stephen Sugden <[me@stephensugden.com](mailto:me@stephensugden.com)> (stephensugden.com)

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

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

mime

The MIT License (MIT)

Copyright (c) 2010 Benjamin Thomas, Robert Kieffer

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

# PrizmDoc v12.3 - Updated June 23, 2017 48

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

mkdirp

The MIT License (MIT)

Copyright (c) 2010 James Halliday ([mail@substack.net](mailto: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>

Mono

Copyright (c) 2001, 2002, 2003 Ximian, Inc and the individuals listed on the ChangeLog entries.

Download: <http://www.go-mono.com/mono-downloads/download.html>

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

mv

The MIT License (MIT)

Copyright (c) 2014 Andrew Kelley

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

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

netty-buffer

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>

netty-codec

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>

netty-codec-http

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>

netty-common

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>

netty-handler

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>

# PrizmDoc v12.3 - Updated June 23, 2017 49

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/NLog/LICENSE.txt

(<https://github.com/NLog/NLog/blob/master/LICENSE.txt>)

NLog

Copyright (c) 2004-2011 Jaroslaw Kowalski

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/blob/master/LICENSE.txt>

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-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-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](mailto:felix@debuggable.com)) and contributors

Download: <https://github.com/felixge/node-mysql>

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

node-remove

The MIT License (MIT)

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

# PrizmDoc v12.3 - Updated June 23, 2017 50

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

node-retry

Copyright: (c) 2011:

Tim Koschützki ([tim@debuggable.com](mailto:tim@debuggable.com))

Felix Geisendörfer ([felix@debuggable.com](mailto:felix@debuggable.com))

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

license: MIT <https://github.com/tim-kos/node-retry/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](mailto: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

Copyright (C) Joyent, Inc. and other Node contributors

Download: <http://nodejs.org/dist/v0.10.26/>

Version: 0.10.26

License: <install directory>\licenses\Node.js\LICENSE.txt

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://sourceforge.net/projects/openjpeg.mirror/files/>

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

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

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/>)

# PrizmDoc v12.3 - Updated June 23, 2017 51

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>

Poppler

Copyright (c) 2005-2014 The Poppler Developers

Copyright (c) 1996-2011 Glyph & Cog, LLC

Download: <http://poppler.freedesktop.org>

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

## PrizmDoc Server Fonts:

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

Copyright (c) by Font AC

Download: [http://www.ac-font.com/jp/detail\\_jb\\_007.php](http://www.ac-font.com/jp/detail_jb_007.php)

License: <http://www.ac-font.com/jp/terms.php>

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

Copyright (c) 2012, Vicente Lamonaca ([produccion.taller@gmail.com](mailto:produccion.taller@gmail.com))

Download: <https://www.fontsquirrel.com/fonts/economica>

License: <http://scripts.sil.org/OFL> (SIL Open Font License v1.10)

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)

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 v1.10)

Microsoft TrueType core fonts for the Web

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

License: <https://www.microsoft.com/typography/fontpack/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 v1.10)

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)

# PrizmDoc v12.3 - Updated June 23, 2017 52

License)

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](mailto:miloyip@gmail.com))

Download: <https://github.com/miloyip/rapidjson>

Version: v0.11

License: <install directory>/licenses/rapidjson/license.txt

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/rtfparkerkit>)

Copyright (c) 2013 Jon Iles

Download: <https://github.com/joniles/rtfparkerkit>

License: Apache License Version 2.0 (<install directory>/licenses/RTF Parser Kit/licence.txt)

sails-mysql

The MIT License (MIT)

Copyright (c) 2016 Mike McNeil, Balderdash & contributors

Download: <https://github.com/balderdashy/sails-mysql>

License: <install directory>/licenses/sails-mysql/README.md and <https://sails.mit-license.org/>

# PrizmDoc v12.3 - Updated June 23, 2017 53

sails-sqlserver

The MIT License (MIT)

Download: <https://github.com/cnect/sails-sqlserver>

License: <https://github.com/cnect/sails-sqlserver>

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

StringSearch - high-performance pattern matching algorithms in Java

Copyright (c) 2003-2010 Johann Burkard

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

License: <http://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](http://bouncycastle.org/latest_releases.html)

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

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>

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

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

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

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

tunnel-agent

Apache License Version 2.0

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

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

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>

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](http://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](mailto:support@accusoft.com) about the Qt source code distribution with modifications by Accusoft.

wrappy

# PrizmDoc v12.3 - Updated June 23, 2017 54

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 <[aineros@gmail.com](mailto:aineros@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

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

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

yargs

Copyright (c) 2010 James Halliday ([mail@substack.net](mailto: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

## Deprecated Viewers

The following third party software may be used or distributed in the deprecated client components included with the Program:

GraphicsUtil (<http://www.dncompute.com/blog/2008/07/17/graphicsutil-a-utility-class-for-drawing-arrows.html>)

Copyright (c) 2008 Noel Billig ([www.dncompute.com](http://www.dncompute.com))

Download: <http://www.dncompute.com/blog/2008/07/17/graphicsutil-a-utility-class-for-drawing-arrows.html>

License: MIT (in source)

URL Validator (<https://gist.github.com/martinmenneske/5756597>)

Copyright (c) 2009 Martin Jacobsen

Download: <https://gist.github.com/martinmenneske/5756597>

License: (in source)

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

# PrizmDoc v12.3 - Updated June 23, 2017 55

Copyright (c) 2011 John Resig

Download: [http://docs.jquery.com/Downloading\\_jQuery](http://docs.jquery.com/Downloading_jQuery)

License: <http://jquery.org/license/>

Lazy Load - jQuery plugin for lazy loading images (<http://www.appelsiini.net/projects/lazyload>)

Copyright (c) 2007-2009 Mika Tuupola

Download: <http://www.appelsiini.net/projects/lazyload>

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

jQuery.ScrollTo (<https://github.com/flesler/jquery.scrollTo>)

Copyright (c) 2007-2009 Ariel Flesler

Download: <https://github.com/flesler/jquery.scrollTo>

License: MIT (in source)

ScrollView - jQuery plugin (<http://code.google.com/p/jquery-scrollview/>)

Copyright (c) 2009 Toshimitsu Takahashi

Download: [http://code.google.com/p/jquery-scrollview/downloads/detail?](http://code.google.com/p/jquery-scrollview/downloads/detail?name=jquery.scrollview.js&can=2&q=)

[name=jquery.scrollview.js&can=2&q=](http://code.google.com/p/jquery-scrollview/downloads/detail?name=jquery.scrollview.js&can=2&q=)

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

Viewport - jQuery selectors for finding elements in viewport (<http://www.appelsiini.net/projects/viewport>)

Copyright (c) 2008-2009 Mika Tuupola

Download: <http://www.appelsiini.net/projects/viewport>

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

transform: A jQuery cssHooks adding cross-browser 2d transform capabilities to \$.fn.css() and \$.fn.animate()

(<https://github.com/louisremi/jquery.transform.js>)

Copyright (c) 2011 @louis\_remi

Download: <https://github.com/louisremi/jquery.transform.js>

License: MIT (<https://github.com/louisremi/jquery.transform.js>)

jQuery UI (<http://jqueryui.com/>)

Copyright (c) 2011 Paul Bakaus

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

License: <http://jqueryui.com/about/>

jquery.waitforimages (<http://alexanderdickson.com/blog/2011/02/a-new-jquery-plugin-2/>)

Copyright (c) 2011 Alex Dickson

Download: <https://github.com/alexanderdickson/waitForImages>

License: <https://github.com/alexanderdickson/waitForImages/blob/master/README.md>

jQuery Context Menu Plugin (<http://swisnl.github.io/jQuery-contextMenu/index.html>)

Copyright (c) A Beautiful Site, LLC

Download: <https://github.com/swisnl/jQuery-contextMenu>

License: MIT (in source)

jqprint (<https://gist.github.com/sumutcan/2190391>)

Provided by Eros Fratini - [eros@recoding.it](mailto:eros@recoding.it)

Download: <https://gist.github.com/sumutcan/2190391>

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

jQuery Patch (<http://www.zachstronaut.com/posts/2009/08/07/jquery-animate-css-rotate-scale.html>)

2009-2010 Zachary Johnson [www.zachstronaut.com](http://www.zachstronaut.com)

Download: <https://github.com/zachstronaut/jquery-animate-css-rotate-scale/>

# PrizmDoc v12.3 - Updated June 23, 2017 56

License: MIT (<https://github.com/zachstronaut/jquery-animate-css-rotate-scale/blob/master/README>)

## Current Viewer (v9.x and higher)

The following third party software may be used or distributed in the client components included with the Program:

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

Copyright (c) 2014 The jQuery Foundation

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

Version: 1.10.2

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

jQuery Hotkeys

Copyright (c) 2013 by John Resig

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

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

jQuery-Mask-Plugin

The MIT License (MIT)

Copyright (c) 2012 Igor Escobar

Download: <https://github.com/igorescobar/jquery-Mask-Plugin>

License: <https://github.com/igorescobar/jquery-Mask-Plugin/blob/master/LICENSE>

JSDoc 3 (<https://github.com/jsdoc3/jsdoc>)

Copyright (c) 2011-2014 Michael Mathews [micmath@gmail.com](mailto:micmath@gmail.com) and the contributors to JSDoc. All rights reserved.

Download: <https://github.com/jsdoc3/jsdoc>

License: <https://github.com/jsdoc3/jsdoc/blob/master/LICENSE.md>

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

Licensed under MIT license

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

Version: 3.0.0

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

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: <http://opensource.org/licenses/mit-license.php>, <http://www.gnu.org/licenses/gpl-2.0.html>

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

Copyright (c) 2009-2014 Jeremy Ashkenas, DocumentCloud and Investigative Reporters & Editors

Underscore may be freely distributed under the MIT license.

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

Version: 1.6.0

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

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](http://www.kimberlygeswein.com))

Dancing Script:

# PrizmDoc v12.3 - Updated June 23, 2017 57

Copyright (c) 2010, Pablo Impallari ([www.impallari.com](http://www.impallari.com)) [impallari@gmail.com](mailto:impallari@gmail.com)

Copyright (c) 2010, Iginio Marini. ([www.ikern.com](http://www.ikern.com)) [mail@iginomarini.com](mailto: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](mailto:astigma@astigmatic.com))

Great Vibes:

Copyright (c) 2012, TypeSETit, LLC ([typesetit@att.net](mailto:typesetit@att.net))

Italianno:

Copyright (c) 2011, TypeSETit, LLC ([typesetit@att.net](mailto:typesetit@att.net))

La Belle Aurore:

Copyright (c) 2010, Kimberly Geswein ([www.kimberlygeswein.com](http://www.kimberlygeswein.com))

Pacifico:

Copyright (c) 2011, Vernon Adams ([vern@newtypography.co.uk](mailto: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](mailto:astigma@astigmatic.com))

## Third Party Notices – Other

The following third party software may be used or distributed in the sample code included with the Program:

as3crypto

Copyright (c) 2007 Henri Torgemane

Download: <https://code.google.com/archive/p/as3crypto/>

License: <http://as3crypto.googlecode.com/svn/trunk/as3crypto/LICENSE.txt>

Google Translate API for .NET (<http://code.google.com/p/google-api-for-dotnet/>)

Copyright (c) 2008-2009 iron9light

Download: <http://code.google.com/p/google-api-for-dotnet/downloads/list>

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

Other Terms: <https://cloud.google.com/translate/v2/terms>

JSP Sample

GSON library: Son 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>

SWFObject (<https://github.com/swfobject/swfobject>)

Provided by code.google.com

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

# PrizmDoc v12.3 - Updated June 23, 2017 58

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

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>

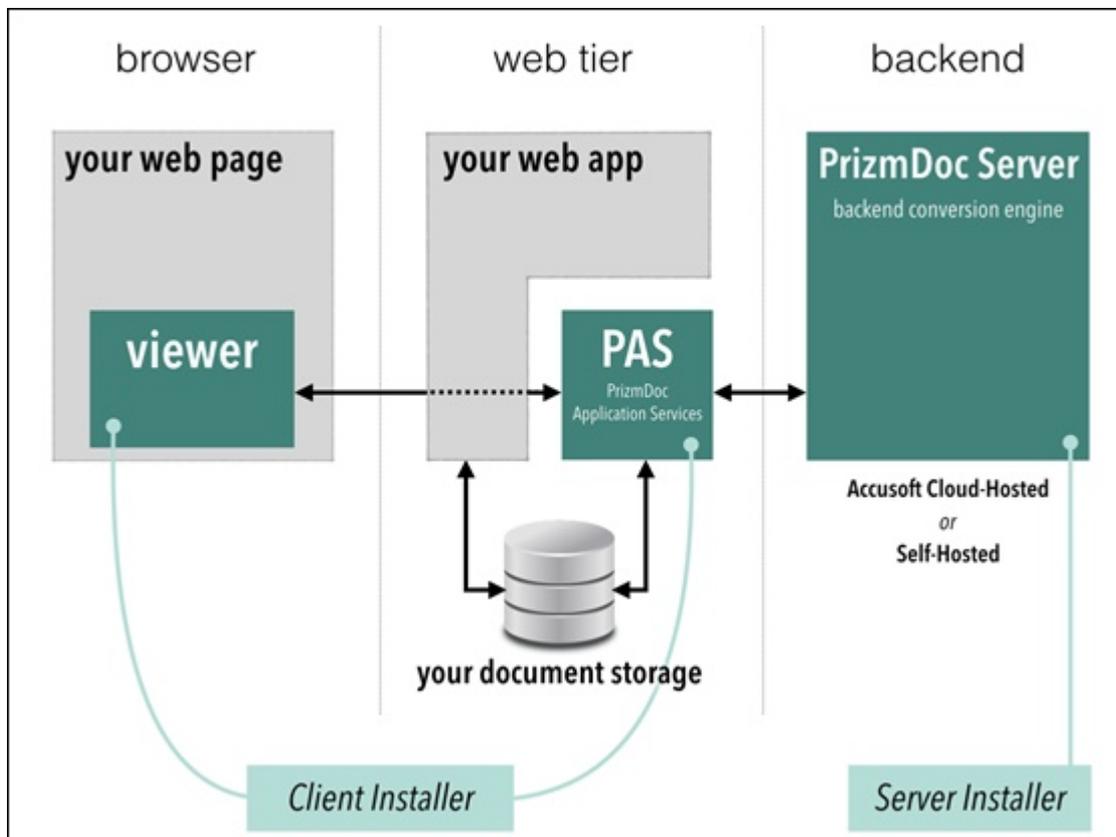
## Get Started with PrizmDoc

Welcome to PrizmDoc; let's get started! This topic walks you through the steps you need to get PrizmDoc installed, configured, and running. You will also be able to integrate a basic viewer into your web app.

It's important to understand the basic structure of PrizmDoc when it's embedded in your web app. PrizmDoc contains 3 essential pieces:

1. **PrizmDoc Server** - [Self-Hosted](#) or [Accusoft Cloud-Hosted](#) - the back-end conversion engine
2. **PrizmDoc Application Services (PAS)** - the [connection point](#) between the Viewer and the Server
3. **PrizmDoc Viewer** - the [HTML viewer component](#)

Example:



Key for diagram above:

- Gray = Your browser, website, and data
- Teal = PrizmDoc components

### Step 1 - Set up a Back-end

In this step, select the back-end that makes the most sense for your needs. The easiest way to get up and running for an evaluation is to select the Accusoft Cloud-Hosted back-end. You can always start with Accusoft Cloud-Hosted for the evaluation and then move to Self-Hosted later. For more details in determining which hosting solution to use, refer to [Server Hosting Options](#).

# PrizmDoc v12.3 - Updated June 23, 2017 60

## Accusoft Cloud-Hosted

- [Get an Evaluation API Key](#)

## Self-Hosted

- [Get an Evaluation License](#)
- [Install PrizmDoc Server](#) (use the *Server Installer*)
- [Check the Back-end Server Health](#)

## Step 2 - Install Viewer Assets & PAS

This step covers installing the Viewer Assets and PrizmDoc Application Services (PAS). After you complete the installation, you need to verify that the Viewer is connected to the back-end.

## Accusoft Cloud-Hosted

- [Install Viewer Assets & PAS](#)
- [Check the PAS Connection to Accusoft Cloud-Hosted Services](#)

## Self-Hosted

- [Install Viewer Assets & PAS](#)
- [Check the PAS Connection to your Back-end Server](#)

## Now, try out a sample:

- If you are on Windows, you can now run our sample .NET web application:

**[http://localhost:18000/PrizmDoc\\_HTML5\\_Viewer\\_NET\\_WEBFORMS/](http://localhost:18000/PrizmDoc_HTML5_Viewer_NET_WEBFORMS/)**

- If you are Linux, you can use our [PHP](#) and [JSP](#) samples which you can optionally configure.

## Step 3 - Integrate the Viewer with Your Application

The following sections help you create a Viewing Session, embed the Viewer, and integrate your web application with PAS:

- [How to Configure PAS in Your Server's Entry Point](#)
- [Create a Viewing Session](#)
- [Embed the Viewer](#)
- [Integrate your Web Application with PAS](#) - This is for evaluation only.
  - [.NET WebForms](#)
  - [.NET MVC 5](#)
  - [PHP](#)
  - [JSP](#)

## Resources

The following additional resources are available:

- [Online Code Examples](#)
- [Online API Code Examples](#)
- [Online Demos](#)
- [Online Videos](#)

# PrizmDoc v12.3 - Updated June 23, 2017 61

## Support

- [Troubleshooting](#)
- [FAQs](#)
- [Contact Support](#)

## 1 - Set up a Back-end

This section will help you get an evaluation license based on your choice of:

- [Accusoft Cloud-Hosted](#)
- [Self-Hosted](#)

## Accusoft Cloud-Hosted

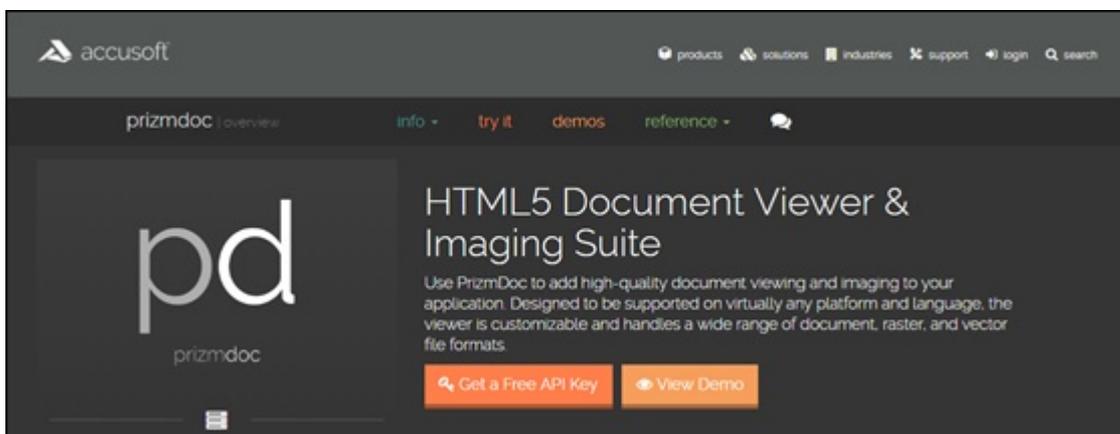
This section will help you get an evaluation API key to get started and show you how to upgrade later:

- [Get an Evaluation API Key](#)
- [Upgrade Your Subscription](#)

## How to Get an Evaluation License

When evaluating PrizmDoc using an Accusoft Cloud-Hosted PrizmDoc Server, you will need an Accusoft Cloud-Hosted Services API key to connect to the back-end service:

1. Open a **browser** and go to <https://www.accusoft.com/products/prizmdoc/overview/> and click **Get a Free API Key**:



2. Fill out the form and click **Get Started**:

# PrizmDoc v12.3 - Updated June 23, 2017 62

The screenshot shows the 'Try PrizmDoc today' form on the Accusoft website. The form includes a list of instructions on the left and a registration form on the right. The instructions are:

- fill out the form**: Fill out the form to instantly start your PrizmDoc trial. We will also send you an email with download information and user guides.
- start your free trial**: Download PrizmDoc to evaluate. If you have any questions, ask us!
- view our pricing options**: Choose the right plan to get you started. Purchase PrizmDoc Cloud-Hosted or Self-Hosted online through our [Buy Now](#) page.

The registration form on the right includes the following fields:

- \* Denotes Required Field
- \* First Name: [Text Input]
- \* Last Name: [Text Input]
- \* Email: A valid email address is required for evaluation license. [Text Input]
- Phone: (optional) [Text Input]
- \* Most Applicable Region: [Dropdown Menu, currently showing 'US']

A 'GET STARTED' button is located at the bottom of the form.

3. Under the **PrizmDoc Cloud** heading, click **Copy to Clipboard** to copy your **API key**:

The screenshot shows the 'PrizmDoc Cloud' and 'PrizmDoc Server' installation instructions page. The 'PrizmDoc Cloud' section includes the following information:

- PrizmDoc Cloud (File Host)**: You will need to download the client installer to run PrizmDoc in the cloud. Your unique API key is provided below.
- Cloud Installation Instructions**: A link to the instructions.
- Your API Key**: `W1uTz43pp3k0Bkrr5pFXB6ZAD0XfDYz4fsm35evygvUv1ZWFq1CpaARxhmZFMZDYE`
- Copy to Clipboard**: A button to copy the API key.
- Download Client**: A button to download the client installer.

The 'PrizmDoc Server' section includes the following information:

- PrizmDoc Server (You Host)**: You will need to download both the server and client installers to run PrizmDoc on your own server.
- Server Installation Instructions**: A link to the instructions.
- Windows**: Supported Server 2008 R SP1 (64-bit), Server 2012 R2, 7 (64-bit), 8 (64-bit), 10 (64-bit). Includes **Download Client** and **Download Server** buttons.
- Ubuntu and Debian x64**: Supported Ubuntu 12.04 LTS, 13.04 (64-bit), 14.04 LTS (64-bit), 15.04 (64-bit); Supported Debian 7 (64-bit). Includes **Download Client** and **Download Server** buttons.
- Red Hat & CentOS x64 (RHEL 7)**: Supported Redhat Enterprise Linux 7 (64-bit); Supported CentOS 7 (64-bit). Includes **Download Client** and **Download Server** buttons.
- Red Hat & CentOS x64 (RHEL 6 & 6)**: Supported Redhat Enterprise Linux 6 (64-bit); Supported CentOS 6 (64-bit). Includes **Download Client** and **Download Server** buttons.
- Generic Linux x64**: Includes **Download Client** and **Download Server** buttons.

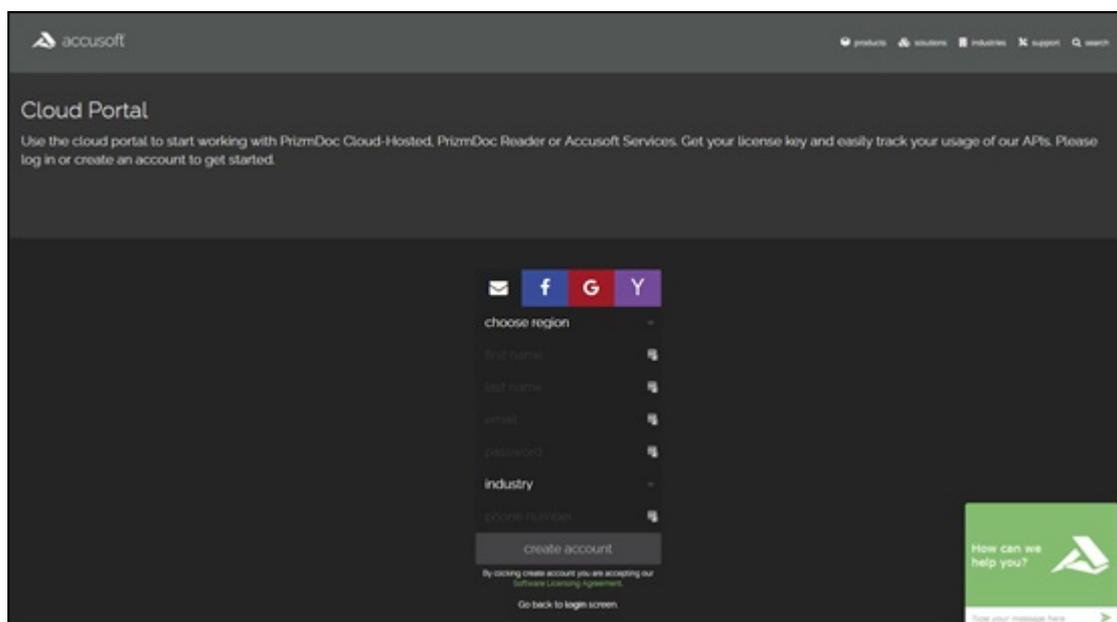
# PrizmDoc v12.3 - Updated June 23, 2017 63

4. Find the operating system you are using and click **Download Client**. (Note that during installation you will be prompted to paste your **API key** into the API key field.)
5. Now go to [Step 2 - Install the Viewer & Set up PAS](#).

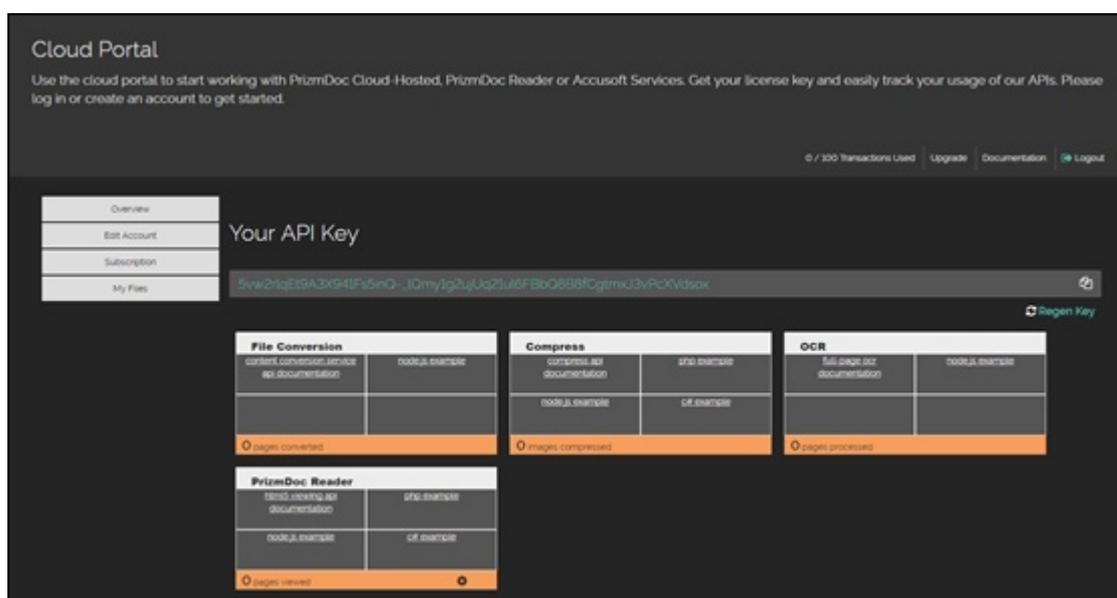
## Upgrade Your Subscription

If you have completed evaluating PrizmDoc and want to upgrade your subscription plan, you can do this through the Cloud Portal without having to make any changes to your code or configuration files:

1. Open a **browser**, go to <https://www.accusoft.com/portal/#> and **log in**:



2. From the Cloud Portal, click on **Upgrade**:



# PrizmDoc v12.3 - Updated June 23, 2017 64

3. Choose the plan that best fits your needs and click **Upgrade**:

The screenshot shows the 'Cloud Portal' interface. At the top, it says 'Use the cloud portal to start working with PrizmDoc Cloud-Hosted, PrizmDoc Reader or Accusoft Services. Get your license key and easily track your usage of our APIs. Please log in or create an account to get started.' Below this, there are navigation links: '0 / 100 Transactions Used', 'Upgrade', 'Documentation', and 'Logout'. A sidebar on the left contains links for 'Overview', 'Edit Account', 'Subscription', and 'My Files'. The main content is a table of subscription plans.

Subscription Type	Transactions	Compressions	Max File Size	Monthly Fee	Current Plan
basic	100	1000	10MB	FREE	Current Plan
freelancer	200	2000	15MB	\$7/mo	Upgrade
professional	1000	10000	20MB	\$49/mo	Upgrade
business	2000	20000	20MB	\$88/mo	Upgrade
business-elite	4000	40000	20MB	\$149/mo	Upgrade
enterprise-essentials	25000	250000	UNLIMITED	\$399/mo	Upgrade
enterprise	50000	500000	UNLIMITED	\$699/mo	Upgrade
enterprise-elite	100000	1000000	UNLIMITED	\$899/mo	Upgrade
contact us	UNLIMITED	UNLIMITED	UNLIMITED	CONTACT US	CONTACT US

4. From the Upgrade subscription dialog, enter your payment information and click **Subscribe**:

The screenshot shows a dialog box titled 'Upgrade subscription to freelancer'. It includes the text: 'Includes access to all Accusoft Cloud Services features, 200 transactions, max file size 15 MB.' There are two radio button options: '\$7 per month' (selected) and '\$5 per month (\$60 billed annually)'. Below these is a 'Discount Code' input field. A large white input field for 'Card Number' is present, followed by an 'Expiration Date' field and a 'CW' field. At the bottom left, there is a checkbox for 'I accept the Terms of Service'. At the bottom right, there are two buttons: 'subscribe to freelancer' and 'Cancel'.

5. Your upgraded PrizmDoc subscription plan is now in effect.

## Self-Hosted

This section will help you get a license key and install the PrizmDoc Server:

- [Get an Evaluation License](#)
- [Install PrizmDoc Server](#)

# PrizmDoc v12.3 - Updated June 23, 2017 65

- Install on Windows
  - Windows Requirements & Supported Environments
  - Windows Installation
  - Unattended Install & Uninstall
  - Uninstall PrizmDoc on Windows
- Install on Linux
  - Linux Requirements & Supported Environments
  - Linux Installation
  - Install Asian Fonts
  - Install on a Headless Environment
  - Uninstall PrizmDoc on Linux
- Check PrizmDoc Server Health
- Configure a Cluster

## Get an Evaluation License

When evaluating PrizmDoc using a Self-Hosted PrizmDoc Server, you will need a Self-Hosted API key:

1. Open a **browser**, go to <https://www.accusoft.com/products/prizmdoc/overview/> and click **Get a Free API Key**:



2. Fill out the online form and click **Get Started**:

# PrizmDoc v12.3 - Updated June 23, 2017 66

The screenshot shows the 'Try PrizmDoc today' registration form on the Accusoft website. The form includes a list of instructions on the left and a registration form on the right. The instructions are:

- fill out the form**: Fill out the form to instantly start your PrizmDoc trial. We will also send you an email with download information and user guides.
- start your free trial**: Download PrizmDoc to evaluate. If you have any questions, ask us!
- view our pricing options**: Choose the right plan to get you started. Purchase PrizmDoc Cloud-Hosted or Self-Hosted online through our [Buy Now](#) page.

The registration form on the right includes the following fields:

- \* Denotes Required Field
- \* First Name: [Text Input]
- \* Last Name: [Text Input]
- \* Email: A valid email address is required for evaluation license. [Text Input]
- Phone: (optional) [Text Input]
- \* Most Applicable Region: [Dropdown Menu, currently set to US]

A 'GET STARTED' button is located at the bottom of the form.

3. After you click Get Started, the **Download** page displays:

The screenshot shows the 'Download' page on the Accusoft website. The page is divided into two main sections: 'PrizmDoc Cloud' and 'PrizmDoc Server'.

**PrizmDoc Cloud (File Host)**: This section provides instructions for downloading the client installer to run PrizmDoc in the cloud. It includes a unique API key: `W1uTz43pp3k0Bkrr5pFXB6ZAD0XfDYz4fsm35evygvUv1ZWFq1CpaARxhmZFMZDYE` and a 'Copy to Clipboard' button.

**PrizmDoc Server (You Host)**: This section provides instructions for downloading both the server and client installers to run PrizmDoc on your own server. It lists various operating systems and versions with corresponding 'Download Client' and 'Download Server' buttons:

- Windows**: Supported Server 2008 R SP1 (64-bit), Server 2012 R2, 7 (64-bit), 8 (64-bit), 10 (64-bit)
- Ubuntu and Debian x64**: Supported: Ubuntu 12.04 LTS, 13.04 (64-bit), 14.04 LTS (64-bit), 15.04 LTS (64-bit); Supported: Debian 71 (64-bit)
- Red Hat & CentOS x64 (RHEL 7)**: Supported: Redhat Enterprise Linux 7 (64-bit); Supported: CentOS 7 (64-bit)
- Red Hat & CentOS x64 (RHEL 6 & 6E)**: Supported: Redhat Enterprise Linux 6.0- 6.4- (64-bit); Supported: CentOS 6.0- 6.4- (64-bit)
- Generic Linux x64**

# PrizmDoc v12.3 - Updated June 23, 2017 67

4. Under the **PrizmDoc Server** heading, find the operating system you are using.
5. Click **Download Server**.
6. Follow the [Install PrizmDoc Server](#) instructions. (Note that during installation, you will be prompted to enter the **email address** you used when signing up for the evaluation. This activates your evaluation license.)
7. After you have finished downloading and installing the PrizmDoc Server, go back to the **Download** page (shown above) and click on **Download Client**.
8. Now go to [Step 2 - Install the Viewer & Set up PAS](#).

For more information on evaluation licensing and troubleshooting licensing issues, refer to the topic [Evaluation Licensing](#).

## Install PrizmDoc Server

This section contains the following information:

- [Install on Windows](#)
  - [Windows Requirements & Supported Environments](#)
  - [Windows Installation](#)
  - [Unattended Install & Uninstall](#)
  - [Uninstall PrizmDoc on Windows](#)
- [Install on Linux](#)
  - [Linux Requirements & Supported Environments](#)
  - [Linux Installation](#)
  - [Install Asian Fonts](#)
  - [Install on a Headless Environment](#)
  - [Uninstall PrizmDoc on Linux](#)
- [Check PrizmDoc Server Health](#)
- [Configure a Cluster](#)

## Install on Windows

This section contains the following information:

- [Windows Requirements & Supported Environments](#)
- [Windows Installation](#)
- [Unattended Install & Uninstall](#)
- [Uninstall PrizmDoc on Windows](#)

## Windows Requirements & Supported Environments

This section provides information about the system requirements for PrizmDoc Server when using a Self-Hosted installation:

 PrizmDoc 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

 Product test validation on the following Operating Systems are deprecated in v12.2 and will be discontinued altogether in v13.0:

- Windows 7
- Windows 8
- Windows 10
- Windows Server 2008 R2 SP1
- Windows Server 2012
- Windows Server 2012 R2

### 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 and the PrizmDoc Server are built targeting the .NET 4.0 framework.

#### Microsoft Office Conversion

# PrizmDoc v12.3 - Updated June 23, 2017 69

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 or 2016 Standard Edition (not included in the PrizmDoc Server distribution and licensed separately) and the corresponding Windows Updates.
- **Microsoft XPS Document Writer** printer driver.
- **Ink and Handwriting Services** feature from the Server Manager.

 The installed copy of Microsoft Office **must be activated** in order for PrizmDoc's Microsoft Office Conversion Service to work properly. Not licensed, not activated, an expired or trial version of Microsoft Office will not work with PrizmDoc.

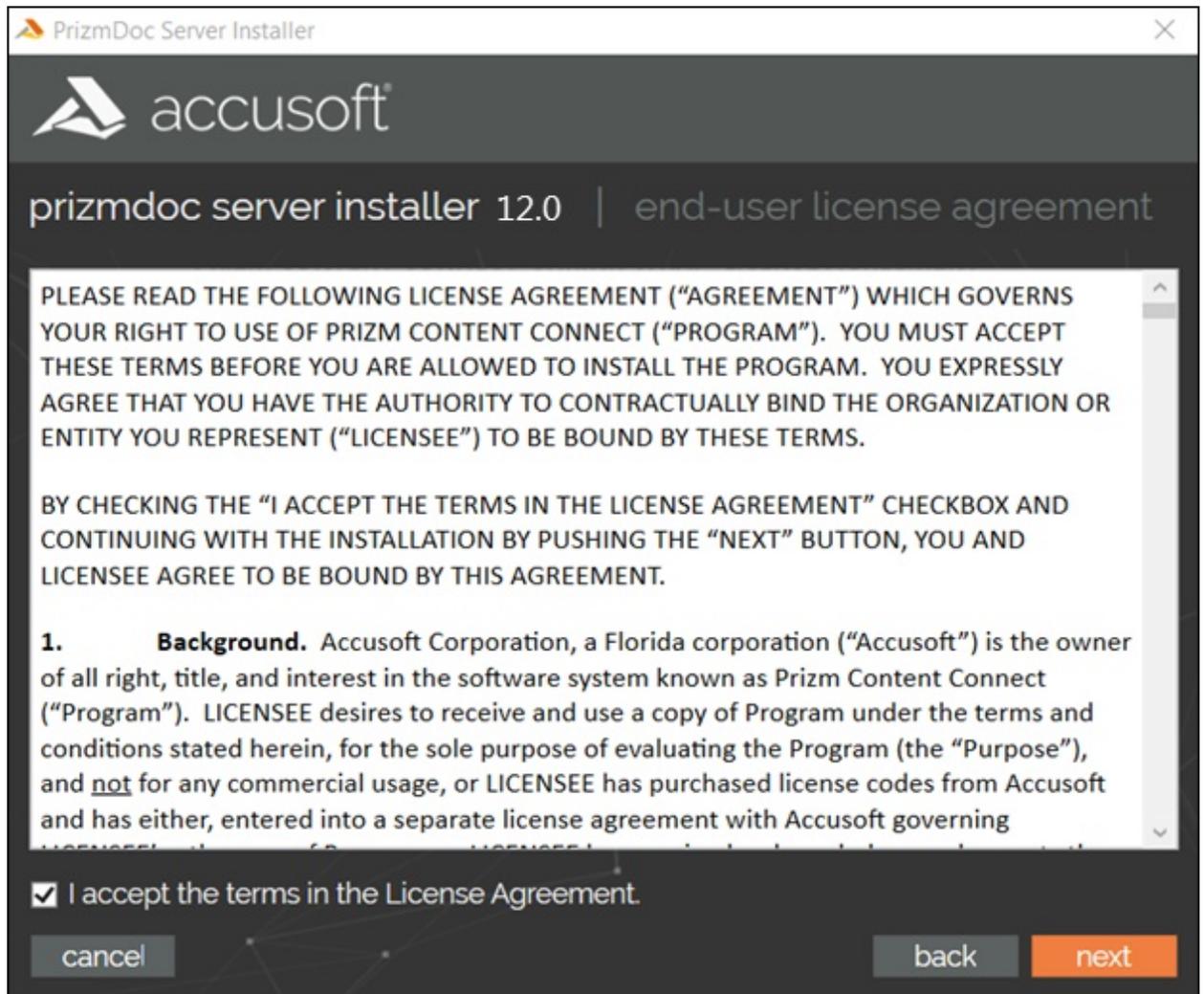
## Windows Installation

To install PrizmDoc Server on your own server, follow these steps:

 PrizmDoc v12.0 requires a clean installation when migrating from an earlier version than 12.0. You must first uninstall any previous versions of PrizmDoc and reboot your system. Only then should you install PrizmDoc v12.0.

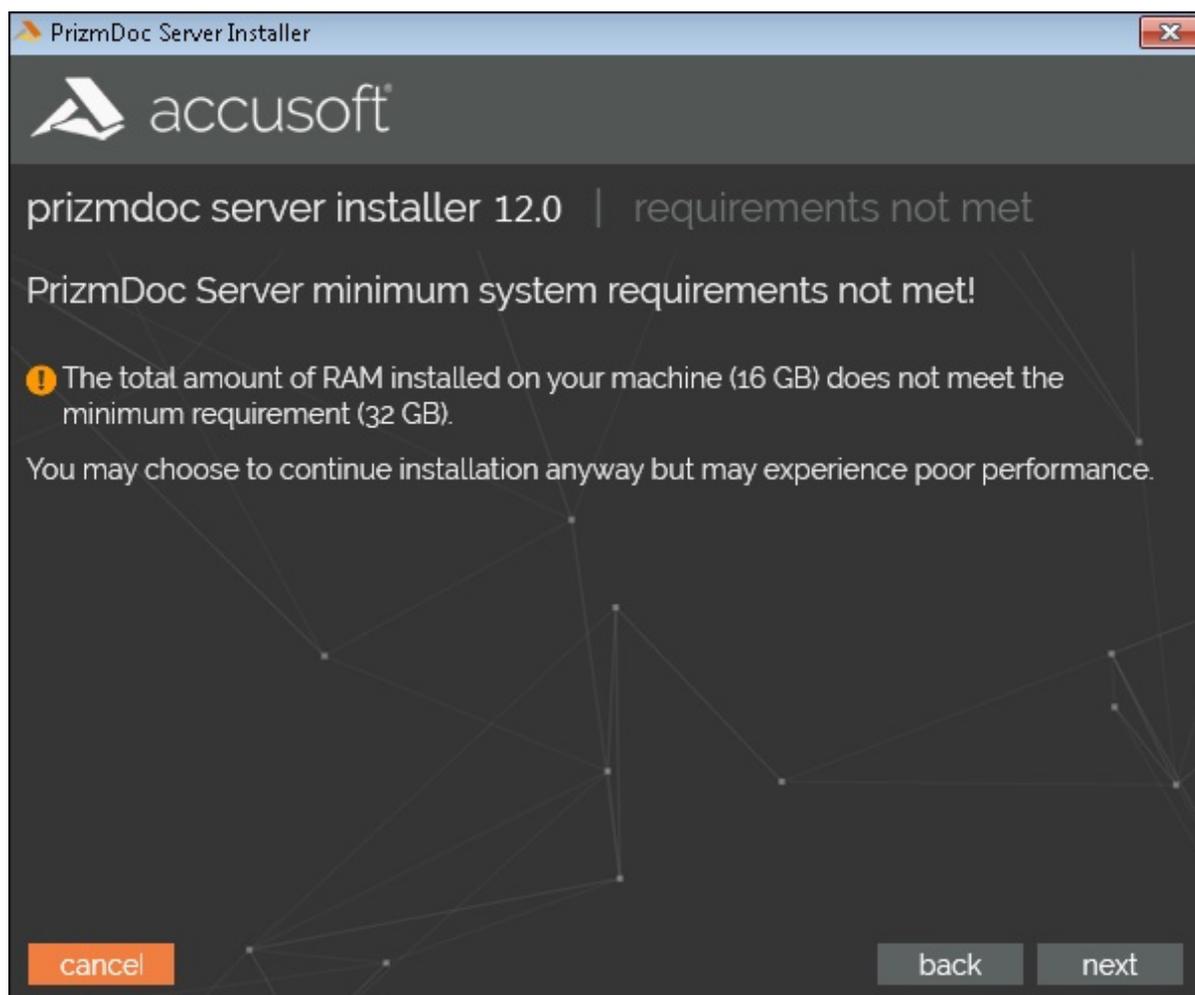
1. Download **PrizmDoc Server** from the [website](#).
2. Carefully read the information contained in the License Agreement form before making a decision to accept the terms of the agreement. Choose **I accept the terms in the License Agreement** to accept the conditions outlined in the License Agreement and then click the **Next** button to continue the installation (or click **Cancel** to exit the installation process):

# PrizmDoc v12.3 - Updated June 23, 2017 70



3. If your machine does not meet the [PrizmDoc Server minimum system requirements](#), a dialog displays indicating the requirements that are not met:

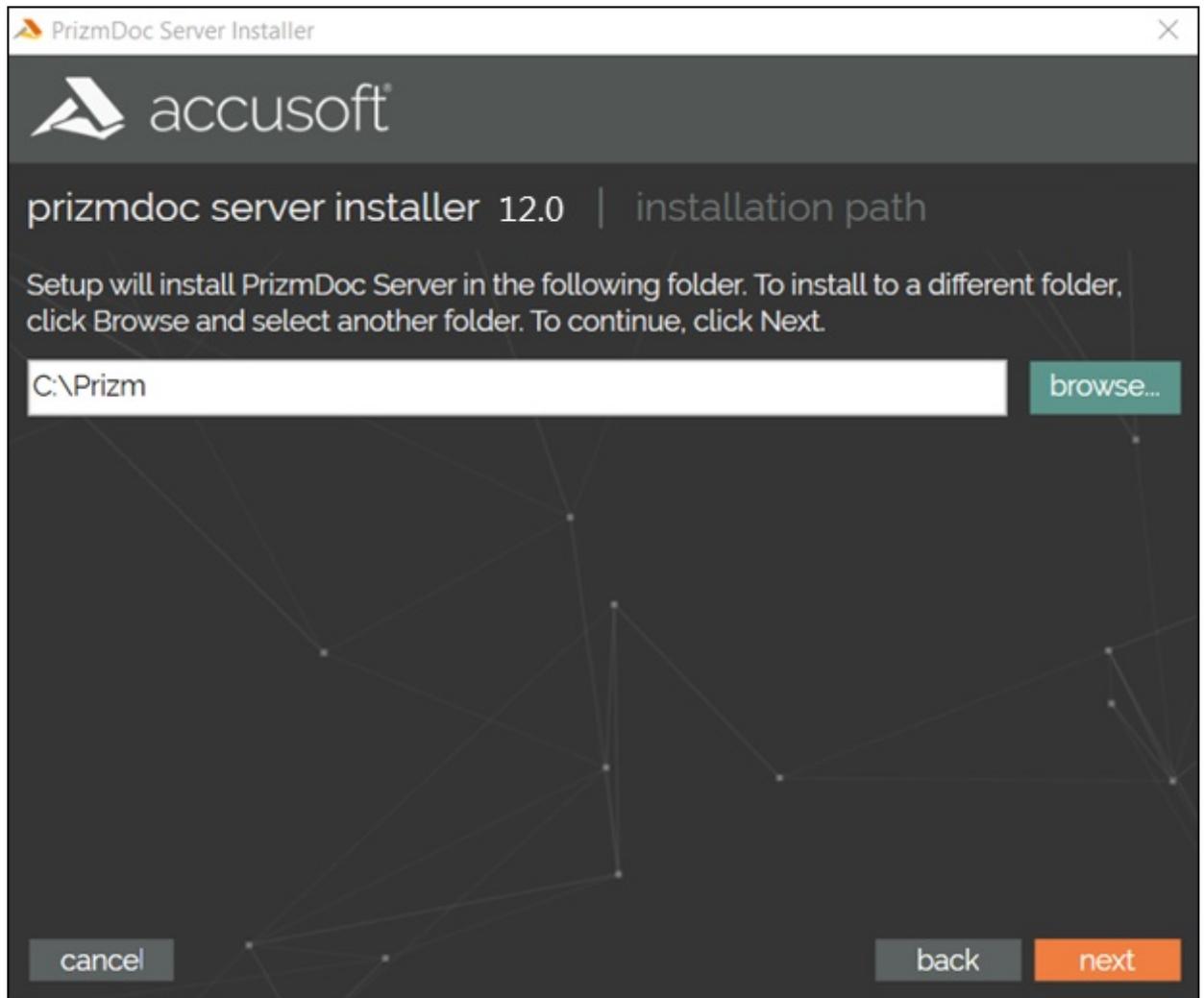
# PrizmDoc v12.3 - Updated June 23, 2017 71



You may choose to continue installation, but may experience poor performance.

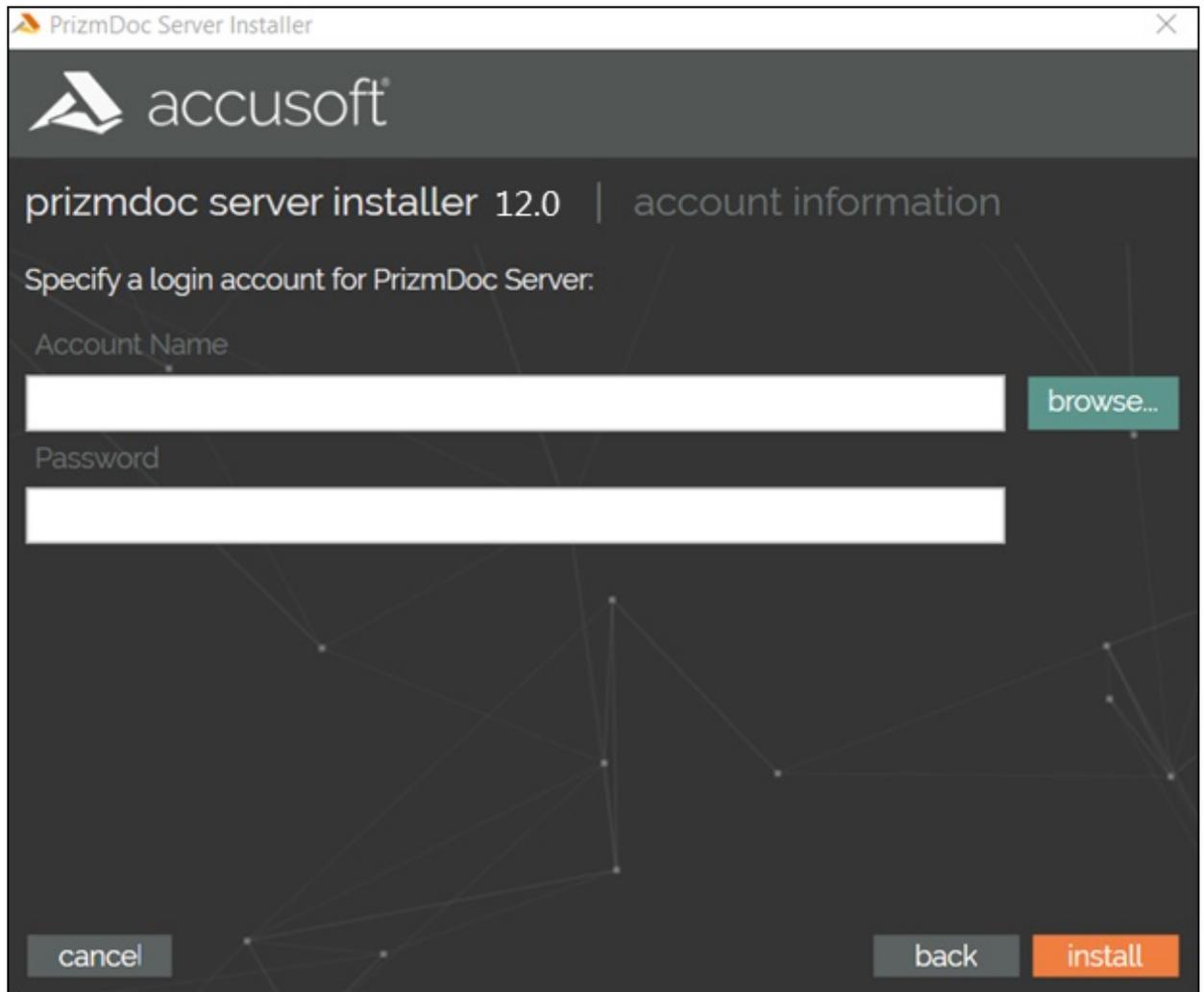
4. The Installation Path dialog is displayed. Specify the **destination directory** where the PrizmDoc product should be installed or choose the default installation destination directory. Then click **Next**:

# PrizmDoc v12.3 - Updated June 23, 2017 72



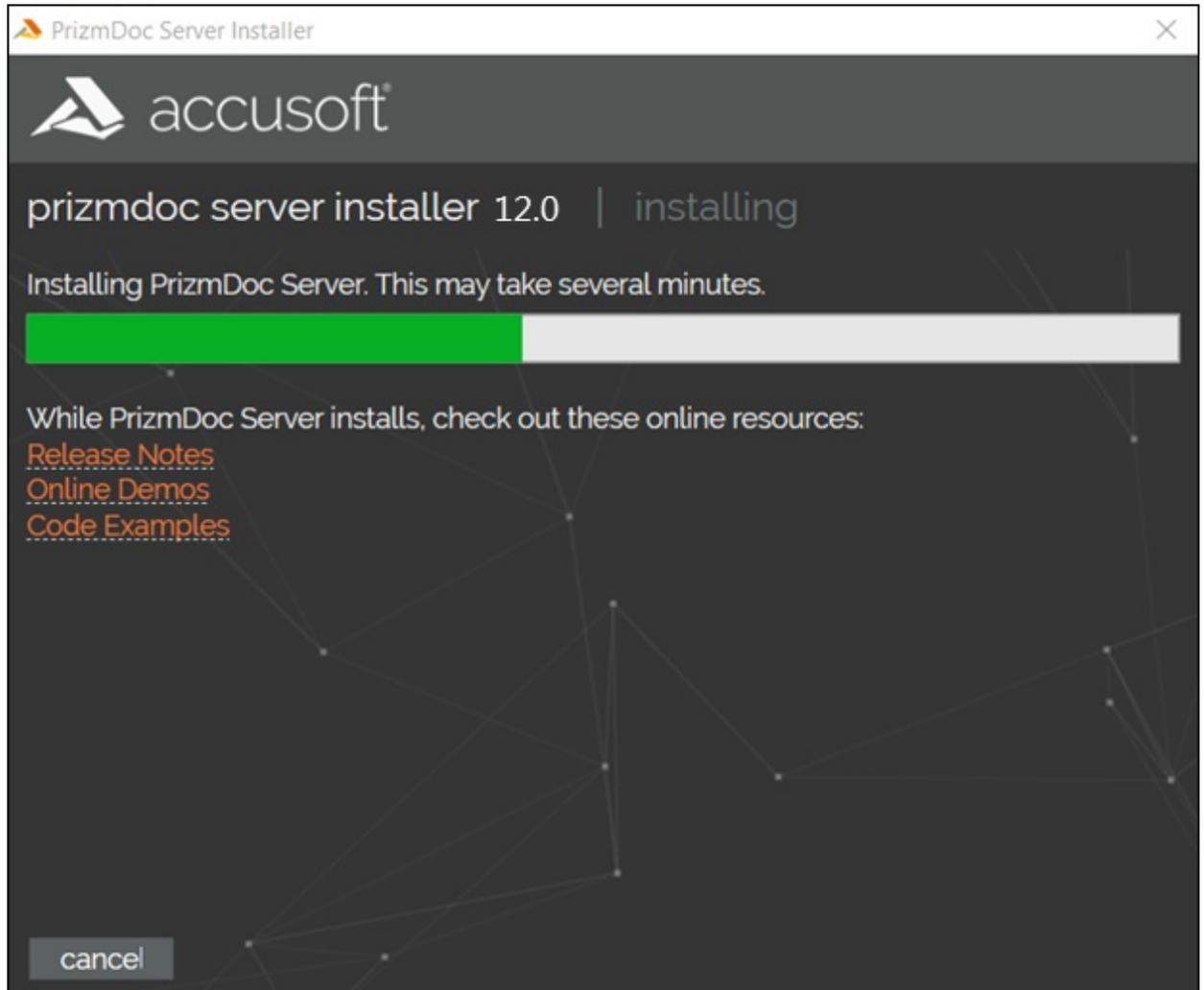
5. Specify the **login account** (account name and password) that PrizmDoc Server will run under. (If you are using the Microsoft Office Conversion (MSO) add-on, please make sure that the "login account" is a real user account with Administrator rights. Running PrizmDoc under the LocalSystem user or another Microsoft Windows integrated service account is not supported for this option):

# PrizmDoc v12.3 - Updated June 23, 2017 73



6. Click **Install** to continue. The Installation dialog is displayed with a progress bar:

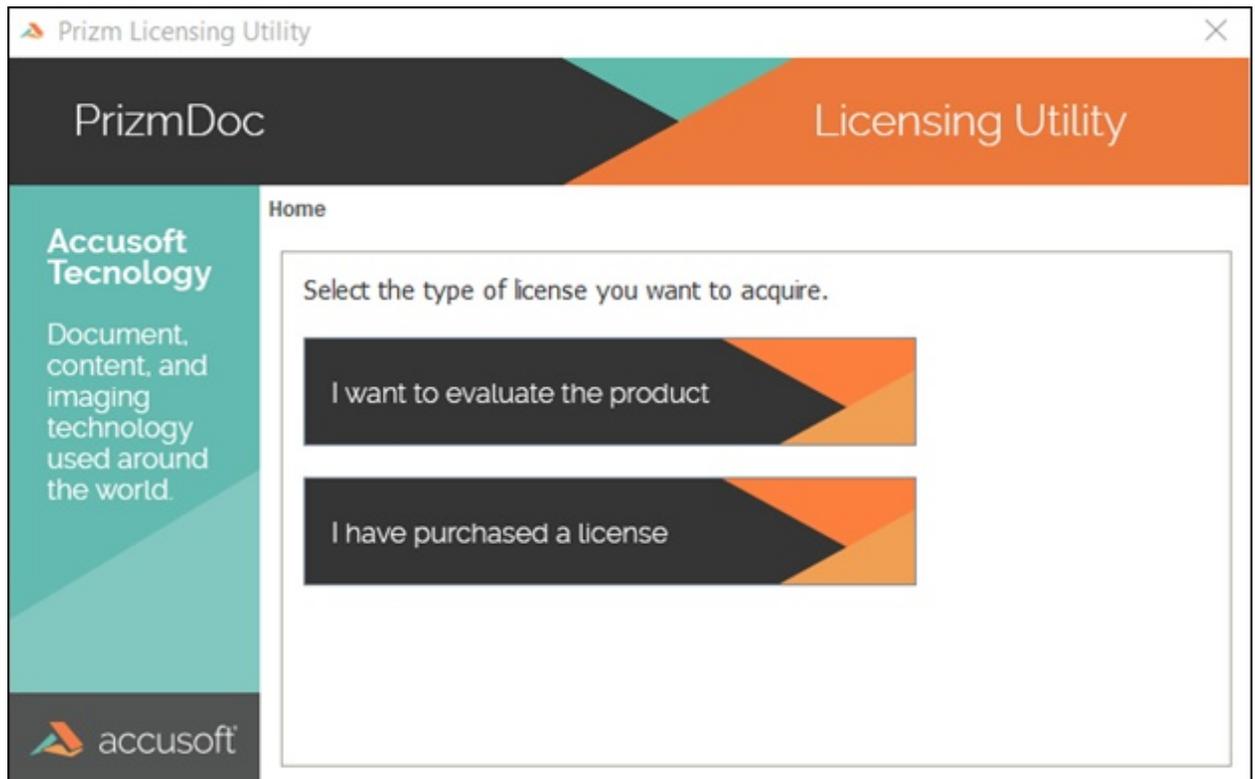
# PrizmDoc v12.3 - Updated June 23, 2017 74



While PrizmDoc Server is installing, you can click on the links to review the **Release Notes**, **Online Demos** or **Code Examples**.

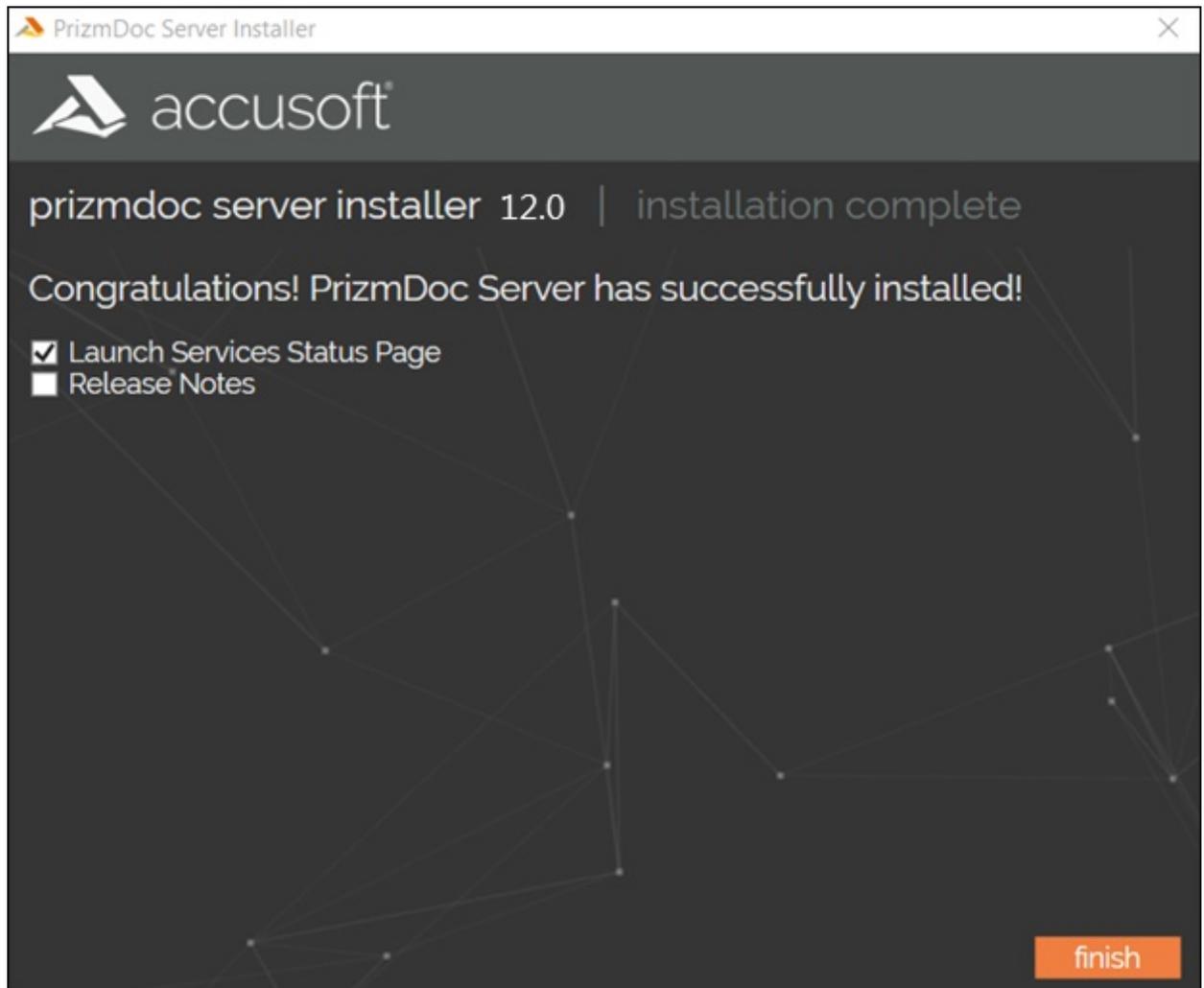
7. Part way through the installation, the installer will launch the **PrizmDoc Licensing Utility**. You can select to **evaluate the product** or enter your license information:

# PrizmDoc v12.3 - Updated June 23, 2017 75



8. After the [PrizmDoc Licensing Utility](#) completes, the installer will continue setting up your system with PrizmDoc Server. Once the configuration is complete, the Installation Complete dialog is displayed:

# PrizmDoc v12.3 - Updated June 23, 2017 76



9. There will be an option to launch PrizmDoc Server Status Page and product Release Notes if you wish to view them directly after the installation. Click **Finish** to exit the Installer. If you selected the option to launch the PrizmDoc Server Status Page, you will see it displayed as shown in the topic: [Checking PrizmDoc Server Health](#).

 For a production installation of PrizmDoc on Windows, you can improve your performance by enabling Gzip compression.

10. Note that Steps 11-14 below are required if you purchased PrizmDoc with the Microsoft Office Conversion service. If you did not purchase PrizmDoc with the Microsoft Office Conversion service, then your installation is complete.
11. The Microsoft Office Conversion Service, that is bundled with PrizmDoc, requires a Windows Registry modification in order to work properly, such as:

**Registry Key: HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\SubSystems**

**Value: Windows**

That value is a complex, space separated set of system-wide settings. One of them is a "SharedSection" comma-separated sub-value with 3 values. Below is a default value for Windows 2008/2012 x64 systems:

# PrizmDoc v12.3 - Updated June 23, 2017 77

**SharedSection= 1024,20480,768**

The third value is a non-interactive desktop heap size in kilobytes. The installer will increase it (if there is a default value) in order to run MS Office in non-interactive mode properly.



- A system reboot will be required in order for the change to have effect.
- 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.
- If the non-interactive desktop heap size is not the default and meets PrizmDoc requirements, it won't be modified, and the system reboot won't be required. You can check installer's logs to see whether a system reboot was scheduled.

Upon an un-installation of the product, the original non-interactive desktop heap size will be restored, and a system reboot would be required again.

12. The Microsoft Office Conversion Service requires the Microsoft XPS Document Writer printer driver to be installed for the best conversion performance and rendering fidelity of MS Excel documents. The PrizmDoc's installation process sets the Microsoft XPS Document Writer to be the default printer driver on the system. Note that if you need to re-install Microsoft XPS Document Writer, follow the steps in the Microsoft XPS Document section below.
13. Specifically for Windows Server systems, it is important to note that the Microsoft PowerPoint rendering support within Microsoft Office requires you to enable the "**Ink and Handwriting Services**" desktop experience feature from the **Server Manager**.



The installed copy of Microsoft Office **must be activated** in order for PrizmDoc's Microsoft Office Conversion Service to work properly. Not licensed, not activated, an expired or trial version of Microsoft Office will not work with PrizmDoc.

14. 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.

## Microsoft XPS Document Writer

Please note that Microsoft XPS Document Writer comes pre-installed on most Windows systems. However, if the Microsoft XPS Document Writer is not installed on your system, or in case it was previously removed, it must be installed back on the system using the following steps:

1. Click **Start** and select **Devices and Printers**.
2. Click **Add Printer**.
3. Choose **Add a Local Printer**.
4. Select **Use an Existing Port**.
5. Select **XPS Port** and click **Next**.
6. In the manufacturer list, choose **Microsoft**.
7. On the right side, choose the latest version of the XPS document writer and click **Next**.
8. Choose "**Use the driver that is currently installed**", click **Next**.
9. Select the option "**Do Not Share Printer**", then click **Next**.

# PrizmDoc v12.3 - Updated June 23, 2017 78

10. Check the box for "**Set as the default printer**", then click **Finish**.

**If the XPS Port is not available, perform the following steps:**

1. Click **Add a Printer**.
2. Choose **Add a Local Printer**.
3. Select **Create a New Port**.
4. Select **Local Port**, then click **Next**.
5. In the "**Enter a Port Name**" field, type in **XPS Port**, then click **OK**.
6. In the Manufacturer list, choose **Microsoft**.
7. On the right side, choose the latest version of the XPS Document Writer, then click **Next**.
8. Choose "**Use the driver that is currently installed**", click **Next**.
9. Printer name should be **Microsoft XPS Document Writer**, then click **Next**.
10. Select the option "**Do Not Share Printer**", then click **Next**.
11. Check the box for "**Set as the default printer**", then click **Finish**.

 In order to achieve the best performance and fidelity, and avoid potential rendering issues with MS Excel documents, it is important to make sure the Microsoft XPS Document Writer is configured as the default printer for the user which is configured to run the PrizmDoc service.

## Unattended Install & Uninstall

### Unattended Install

The PrizmDoc Server Windows installer can be installed unattended, however certain properties must be set:

 PrizmDoc v12.0 requires a clean installation when migrating from an earlier version than 12.0. You must first uninstall any previous versions of PrizmDoc and reboot your system. Only then should you install PrizmDoc v12.0.

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 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
InstallFolder	Optional - The base installation directory for the product.	"C:\Prizm"

To start the unattended install:

# PrizmDoc v12.3 - Updated June 23, 2017 79

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):

```
> PrizmDocServer.exe ServiceUser=accusoft.com\PrizmUser  
ServicePassword=pdpassword -s -l output.log
```

 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.

```
> start /wait PrizmDocServer.exe ServiceUser=accusoft.com\PrizmUser  
ServicePassword=pdpassword -s -l output.log
```

3. After which the PLU can be used to retrieve a license, for example:

## Example

```
> C:\Prizm\jre6\jre6-windows-x86-64\bin\java.exe -jar  
C:\Prizm\plu\plu.jar deploy get C:\path\to\License_Config.txt "License  
Solution Name"
```

 For more details on command line parameters to the licensing utility, refer to the Prizm License Utility section [Command-Line Mode](#).

4. Next, the service needs to be started. See the [Getting Started with PrizmDoc > Starting & Stopping PrizmDoc Server > Windows](#) for more information on stopping and starting the service.

## Example

```
> net start prizm
```

5. PrizmDoc should now be installed, licensed and started.

 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.

## Unattended Uninstall

# PrizmDoc v12.3 - Updated June 23, 2017 80

You can use the -u flag as shown below to silently uninstall PrizmDoc on Windows:

```
> PrizmDocServer.exe -s -u -l output.log
```

## Uninstall PrizmDoc on Windows

To uninstall Prizm from your Windows system, please perform the following steps:

1. Run **Add/Remove Programs** or **Programs and Features** from the Control Panel.
2. Choose **Prizm** 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 Prizm from your system.

 You may be prompted to reboot your system in order to completely remove the PrizmDoc Server.

## Install on Linux

This section contains the following information:

- [Linux Requirements & Supported Environments](#)
- [Linux Installation](#)
- [Install Asian Fonts](#)
- [Install on a Headless Environment](#)
- [Uninstall PrizmDoc on Linux](#)

## Linux Requirements & Supported Environments

This section provides information about the system requirements for PrizmDoc Server when using a Self-Hosted installation:

 PrizmDoc 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

# PrizmDoc v12.3 - Updated June 23, 2017 81

## Linux

 Product test validation on the following Operating Systems are deprecated in v12.2 and will be discontinued altogether in v13.0:

- CentOS 5.9
- Red Hat Enterprise Linux 5.9
- Ubuntu 12.04 LTS, 13.04

- CentOS 6.4, 7
- Debian 7.1 (Wheezy)
- Red Hat Enterprise Linux 6.4, 7
- Ubuntu 14.04 LTS

## 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.

## 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
- libpixmap-1.so.0
- libpng12.so.0
- libpthread.so.0
- librt.so.1
- libstdc++.so.6

# PrizmDoc v12.3 - Updated June 23, 2017 82

- 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

 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 (CentOS 5.x)

- openjpeg-libs-1.3-7.el5.x86\_64.rpm
- pixman-0.22.0-2.2.el5\_10.x86\_64.rpm

## Linux Installation

PrizmDoc installation is designed to be very straightforward for Linux environments. Follow the steps provided in this section.

 PrizmDoc v12 requires a clean installation when migrating from an earlier version than 12. You must first uninstall any previous versions of PrizmDoc and reboot your system. Only then should you install PrizmDoc v12.

- [Verify the System's Locale](#)
- [Step 1 - Download PrizmDoc Server](#)
- [Step 2 - Unpack & Install the Downloaded Archive](#)
- [Step 3 - Configure](#)
- [Step 4 - Verify that the Installation was Successful](#)
- [Working with Sample Code](#)
- [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 / Fedora (CentOS) Linux Distributions"
- "Debian (Ubuntu) Linux Distributions"

 Make sure you log in as **root** to the machine.

# PrizmDoc v12.3 - Updated June 23, 2017 83

## 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 locale-gen
```

## Step 1 - Download PrizmDoc Server

 Before you download PrizmDoc, note that packages are only available for 64-bit systems.

Before downloading PrizmDoc, you will need to purchase a license key or request a Trial Evaluation by filling out the following form: [www.accusoft.com/products/prizm-content-connect-pcc/get-self-hosted/](http://www.accusoft.com/products/prizm-content-connect-pcc/get-self-hosted/).

Once you have purchased a license key or filled out the form for a Trial Evaluation, you can download PrizmDoc by:

1. Following the link provided in the response email and selecting the desired Linux Distribution.

OR

2. Downloading directly to the Linux server using the 'wget' command for the specific distribution as shown below:

 **You must substitute the version of the package you are using in the code examples below.** For example, if you are using v11.0, then specify "11.0" instead of "<version>". If the version is a hot fix, you will need to also specify the hot fix number, for example, "11.0.1".

### Red Hat Enterprise Linux, CentOS (prior to v7) and Similar Older Linux Distributions

#### Example

```
wget
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_server_<version>.x86_64.rpm.tar.gz
```

### Red Hat Enterprise Linux and CentOS v7 (and later)

#### Example

```
wget
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_server_<version>.RHEL7.x86_64.rpm.tar.gz
```

### Debian and 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
```

# PrizmDoc v12.3 - Updated June 23, 2017 84

For license questions, fill out the form at <http://www.accusoft.com/contact/>.

## 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\_server\_<version>.<arch>[.rpm|.deb]**.
2. Change to the unpacked directory and install the packages.

### Red Hat, Fedora, CentOS, and Older Linux Distributions

The following example is for Red Hat, Fedora, CentOS, and older Linux distributions:

#### Example

```
tar -xzvf prizmdoc_server_*.rpm.tar.gz
cd prizmdoc_server_*.rpm
yum install --nogpgcheck *.rpm
```

 Note: For CentOS 5.x, the following additional dependencies **must** be installed prior to PrizmDoc RPM installation:

- openjpeg-libs-1.3-7.el5.x86\_64.rpm
- pixman-0.22.0-2.2.el5\_10.x86\_64.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.el5.x86_64.rpm
yum install --nogpgcheck ./pixman-0.22.0-2.2.el5_10.x86_64.rpm
```

### Debian (Ubuntu) Linux Distributions

The following example is for Debian (Ubuntu) Linux distributions:

#### 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
there are errors about missing dependencies, invoke next command to install dependencies and
complete configuration of the packages.
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 system fonts cache to enable the usage of installed fonts by PrizmDoc

# PrizmDoc v12.3 - Updated June 23, 2017 85

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 - Configure

1. Continuing as **root**, change to the installation location: **/usr/share/prizm** and run **setup.sh**. This will run the **Prizm Licensing Utility (PLU)** and configure PrizmDoc for its first run:

 setup.sh launches the Prizm License Utility (PLU), which is a GUI application. Depending on how you have X11 configured, you might need to run 'xhost +' as the logged in user to allow the PLU launched by the root user to access X11. You can disable X11 access for root by executing 'xhost -' when you are done.

## Example

```
./setup.sh
Select the type of license you want to acquire:
    1) Evaluation [e]
    2) Deployment [d]
Choose 1 or 2: d
Select the type of deployment license you have:
    1) Node Locked [n]
    2) OEM [o]
Choose 1 or 2: o
Provide the solution name: <MySolutionName>
Provide the OEM license key provided to you: 2.0...
Your deployment license was acquired successfully.
./scripts/pccis.sh start
Starting Prizm Content Connect Information Services...
Starting PCCIS Watchdog process...
PCCIS Watchdog has been started correctly.
```

2. For a production installation, you will want to configure where log files are stored. See the section [How to Configure Log File Locations](#).
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](#).

## Step 4 - Verify that the Installation was Successful

1. Open the PrizmDoc Services Status Page as described in the [Checking PrizmDoc Server Health](#) topic and verify the PrizmDoc Services are running.
2. Your installation is now complete.

## Working with Sample Code

Sample code is included to demonstrate how PrizmDoc can be integrated into your Content Management solution. Sample applications using different languages are packaged with the product to demonstrate using a particular language. All of the product samples are located in the "Samples" folder within the installation directory. For more information, refer to the [Code Samples](#) topic.

## 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.

# PrizmDoc v12.3 - Updated June 23, 2017 86

1. Install all **Mozilla's root certificates**. This command will install all the root certificates Mozilla accepts, prompting only to remove certificates that Mozilla doesn't list, adding all it does trust without prompt. This should make HTTPS work for any system that Firefox works with:

## Example

```
/usr/share/prizm/mono/64/bin/mozroots --machine --import --ask-remove
```

2. Pull **certificates** from an https connection to accept:

## Example

```
/usr/share/prizm/mono/64/bin/certmgr -ssl -m https://servertoadd/
```

As long as the server includes the entire chain to root, this will allow the certificate to authenticate. This will generally be the case for self-signed test certificates. For more complex situations, any required items from the chain will need to be added manually.

3. Manually add the **certificate** from the file:

## Example

```
/usr/share/prizm/mono/64/bin/certmgr -add -c -m /path/to/certificatefile
```

The certificate stores the certificates at: /usr/share/mono/certs/.

## Install Asian Fonts

Important information for installing Asian fonts:

- [Red Hat & CentOS](#)
- [Ubuntu & Debian](#)

### Red Hat & CentOS

#### Version 6

In the RHEL 6 / CentOS 6 release, the language font group packages are no longer available. Instead, these releases have "groupinstall" lists which bundle multiple packages together. In order to see all language packages available through this interface, you can type the following command:

## Example

```
# yum grouplist
```

This will output all group packages, the last section of which is "Available Language Groups". Note that some packs may already be installed, in which case, they will be visible under "Installed Language Packs" at the top of the list. In this case, "chinese-fonts" is now "Chinese Support".

To install this package, run the following command:

## Example

```
# yum groupinstall "Chinese Support"
```

To install all languages listed in the help file, run the following commands:

## Example

# PrizmDoc v12.3 - Updated June 23, 2017 87

```
# yum groupinstall "Chinese Support"  
# yum groupinstall "Japanese Support"  
# yum groupinstall "Korean Support"  
# yum groupinstall "Kannada Support"  
# yum groupinstall "Hindi Support"
```

## Version 7

In the RHEL 7 / CentOS7 release, the language font group packages are no longer available. Instead, these releases have "groupinstall" lists which bundle multiple packages together. There is a single Fonts package which includes support for these languages:

### Example

```
# yum groupinstall Fonts
```

## Ubuntu & Debian

By default, Asian language support is not installed on Ubuntu/Debian 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
```



If PrizmDoc was running when you installed language/font support, you must restart PrizmDoc in order to apply the changes.

## Install on a Headless Environment

Use the following steps to install PrizmDoc in a Linux headless environment:

 PrizmDoc v12.0 requires a clean installation when migrating from an earlier version than 12.0. You must first uninstall any previous versions of PrizmDoc and reboot your system. Only then should you install PrizmDoc v12.0.

- [Verify the System's Locale](#)
- [Step 1 - Download PrizmDoc](#)
- [Step 2 - Unpack & Install the Downloaded Archive](#)
- [Step 3 - Setup](#)
- [Step 4 - Running an X Virtual Framebuffer Server](#)
- [Step 5 - 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 locale-gen
```

### Step 1 - Download PrizmDoc

 Before you download PrizmDoc, note that packages are only available for 64-bit systems.

Before downloading PrizmDoc, you will need to purchase a license key or request a Trial Evaluation by filling out the following form: [www.accusoft.com/products/prizm-content-connect-pcc/get-self-hosted/](http://www.accusoft.com/products/prizm-content-connect-pcc/get-self-hosted/).

Once you have purchased a license key or filled out the form for a Trial Evaluation, you can download PrizmDoc by:

1. Following the link provided in the response email and selecting the desired Linux Distribution.

OR

2. Downloading directly to the Linux server using the 'wget' command for the specific distribution as shown below:

 **You must substitute the version of the package you are using in the code examples below.** For example, if you are using v11.0, then specify "11.0" instead of "<version>". If the version is a hot fix, you will need to also specify the hot fix number, for example, "11.0.1".

#### Red Hat Enterprise Linux, CentOS (prior to v7) and Similar Older Linux Distributions

##### Example

```
wget  
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_server_<version>.x86_64.rpm.tar.gz
```

#### Red Hat Enterprise Linux and CentOS v7 (and later)

# PrizmDoc v12.3 - Updated June 23, 2017 89

## Example

```
wget  
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_server_<version>.RHEL7.x86_64.rpm.tar.gz
```

## Debian and Ubuntu Linux Distributions

### Server Example

```
wget  
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_server_<version>.amd64.deb.tar.gz
```

## Generic .tar.gz Distribution

### Server Example

```
wget http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_server_<version>.x86_64.tar.gz
```

For license questions, fill out the form at <http://www.accusoft.com/contact/>.

## 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 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, Fedora, CentOS, and Older Linux Distributions

The following example is for Red Hat, Fedora, CentOS, and older Linux distributions:

### Viewer Example

```
tar -xzvf prizmdoc_client_*.rpm.tar.gz  
cd prizmdoc_client_*.rpm  
yum install --nogpgcheck *.rpm
```

### Server Example

```
tar -xzvf prizmdoc_server_*.rpm.tar.gz  
cd prizmdoc_server_*.rpm  
yum install --nogpgcheck *.rpm
```

 Note: For CentOS 5.x, the following additional dependencies **must** be installed prior to PrizmDoc RPM installation:

- openjpeg-libs-1.3-7.el5.x86\_64.rpm
- pixman-0.22.0-2.2.el5\_10.x86\_64.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.el5.x86_64.rpm  
yum install --nogpgcheck ./pixman-0.22.0-2.2.el5_10.x86_64.rpm
```

## Debian (Ubuntu) Linux Distributions

The following example is for Debian (Ubuntu) Linux distributions:

### Viewer Example

```
tar -xzvf prizmdoc_client_*.deb.tar.gz  
cd prizmdoc_client_*.deb  
sudo dpkg -i *.deb
```

# PrizmDoc v12.3 - Updated June 23, 2017 90

```
# 'dpkg' does not resolve dependencies automatically, so please ignore possible errors, if you
did not install required dependencies yet, and invoke next command
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 command
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/
```

## Step 3 - Setup

1. Run the following commands to initiate setup:

### Example

```
cd /usr/share/prizm
sh ./setup.sh
```

The setup script will now invoke a text-based wizard for licensing:

 To install a license at the end of the evaluation period, (for example, a deployment license), run the following command:

### Example

```
./setup.sh eval get youremail@example.com
```

If you already have a license for PrizmDoc, you can use one of the other licensing options, detailed in the [Command Line Mode Licensing](#) section of the help.

2. To confirm the licensing has succeeded, run the following command, which will output your license key:

### Example

```
cat ./prizm-services-config.yml | grep license.key
> license.key=<long string>
```

3. If PrizmDoc has started and no errors have been reported, then the installation process is complete.

 However, if you have an error in the log stating that it cannot connect to the Office converter Worker-1 instance, then you must continue to the 'Step 4 - Running an X Virtual Framebuffer' section below.

# PrizmDoc v12.3 - Updated June 23, 2017 91

## Step 4 - Running an X Virtual Framebuffer Server

A full version of the X11 server is not required to run PrizmDoc. However, the minimum requirement is the X Virtual Framebuffer Server, which you can install and run separately from an entire X11 instance.

1. To install the **X Virtual Framebuffer Server**, run the following command:

### Debian/Ubuntu

#### Example

```
sudo apt-get install xvfb
```

### Red Hat / CentOS

#### Example

```
yum install xorg-x11-server-Xvfb
```

2. Start the **service** by running the following command, and then export the instance to the DISPLAY environment variable expected by the PrizmDoc Server:

#### Example

```
Xvfb :20 >/dev/null 2>&1 &  
export DISPLAY=:20
```

3. Start **PrizmDoc**:

#### Example

```
/usr/share/prizm/scripts/pccis.sh start
```

#### Required:

The X Virtual Framebuffer Server must be running any time that the PrizmDoc Server is running.

#### Suggested:

It may be appropriate to create a script that will start the Xvfb service and PrizmDoc Server (the 3 lines above) to execute whenever the machine is restarted.

## Step 5 - Verify that the Installation was Successful

2. Open the PrizmDoc Services Status Page as described in the [Checking PrizmDoc Server Health](#) topic and verify the PrizmDoc Services are running.
2. Your installation is now complete.

## Uninstall PrizmDoc on Linux

To uninstall PrizmDoc from your Linux system, perform the following steps:

1. Stop the service, depending upon where you installed. This will usually be:

```
# /usr/share/prizm/scripts/pccis.sh stop
```

2. Next remove the installed files:

# PrizmDoc v12.3 - Updated June 23, 2017 92

Debian/Ubuntu:

```
# sudo apt-get remove prizm-services.*
```

Red Hat/CentOS:

```
# yum remove prizm-services*
```

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
```

This will also remove PrizmDoc Viewer if it is installed.

```
# rm -Rf /usr/share/prizm
```

3. There will potentially be temporary files left behind. The same command to remove the generic package can be used to remove all remaining files. This will also remove PrizmDoc Viewer if it is installed.

```
# rm -Rf /usr/share/prizm
```

## Check PrizmDoc Server Health

### PrizmDoc Services Status Page

You can view the PrizmDoc Services Status page by browsing to one of the following locations on the machine where you installed PrizmDoc Server:

- For single-server mode - <http://server:18681/admin>
- When clustering is enabled - <http://server:18682/admin>

# PrizmDoc v12.3 - Updated June 23, 2017 93

Home Contact



Prizm Services Status **Running**  Auto-Refresh

License Status	licensed as 'PrizmDoc12'
PCC Version	12.3
OS	Unix 3.10.0.514
Start Time	2017-04-11T12:54:26.3143870Z
Instance ID	plt-12

### Internal Services

Service	Status
PCC Imaging Services	✓
PCC Error Reporting Service	✓
PCC Imaging Conversion Service	✓
PCC PDF Processing Service	✓
PCC Raster Conversion Service	✓
PCC Vector Conversion Service	✓
PCC Html Conversion Service	✓
PCC Work File Service	✓
PCC Office Conversion Service	✓
PCC Format Detection Service	✓
PCC AutoRedaction Service	✓
PCC Redaction Service	✓
PCC Email Processing Service	✓
PCC Email Conversion Service	✓
PCC Content Conversion Service	✓
document-conversion-service	✓
form-extraction-service	✓
mongo-manager-service	✓
configuration-service	✓
licensing-service	✓
health-service	✓
process-state-service	✓
raster-form-extraction-service	✓
text-service	✓

 It may take several minutes for PrizmDoc Server to become completely healthy. The page will auto-refresh as PrizmDoc Server comes online.

Also note that 'PCC' means the 'PrizmDoc' services in the example above.

## Configure a Cluster

 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 multi-server 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 Multi-Server Mode works and steps to configure your server for multi-server use, please see our [Multi-Server Mode](#) introduction. If you're an existing customer and already familiar with Multi-Server Mode, you can find information on our [Multi-Server Mode API here](#).

With a multi-server 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 RESTful API's also require special consideration when used with Multi-Server Mode. In Multi-Server, 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 Multi-Server Mode, please see our [Affinity Tokens & Multi-Server Mode](#) topic.

## 2 - Install Viewer Assets & PAS

This section will help you install the PrizmDoc Viewer and set up PAS based on your choice of:

- [Accusoft Cloud-Hosted](#)
- [Self-Hosted](#)

## Accusoft Cloud-Hosted

This section contains the following information:

- [System Requirements & Supported Environments](#)

- [Install PrizmDoc Viewer](#)
  - [Install on Windows](#)
    - [Install the Viewer](#)
    - [Unattended Install & Uninstall](#)
  - [Install on Linux](#)
- [Check the Connection to Accusoft Cloud-Hosted Services](#)

## System Requirements & Supported Environments

This section provides information about the system requirements for PrizmDoc Application Services (PAS) when using an Accusoft Cloud-Hosted installation.

### PrizmDoc Application Services

PrizmDoc Application Services is a suite of RESTful APIs that primarily proxy to PrizmDoc Server. It requires significant network throughput, disk I/O, and very little processing power.

### Supported Operating Systems

#### Windows

- Windows Server 2008 R2 SP1 (64-bit)
- Windows Server 2012
- Windows Server 2012 R2
- Windows 7 (64-bit)
- Windows 8 (64-bit)
- Windows 10 (64-bit)

#### Linux

- CentOS 5.9+, 6.4+, 7 (64-bit)
- Debian 7.1 (Wheezy) (64-bit)
- Red Hat Enterprise Linux 5.9+, 6.4+, 7 (64-bit)
- Ubuntu 12.04 LTS, 13.04 (64-bit), 14.04 LTS (64-bit)

### 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 Application Services uses and how to optimize them for your needs.

## Install PrizmDoc Viewer

# PrizmDoc v12.3 - Updated June 23, 2017 96

This section contains the following information:

- [Install on Windows](#)
  - [Install the Viewer](#)
  - [Unattended Install & Uninstall](#)
- [Install on Linux](#)

## Install on Windows

This section contains the following information:

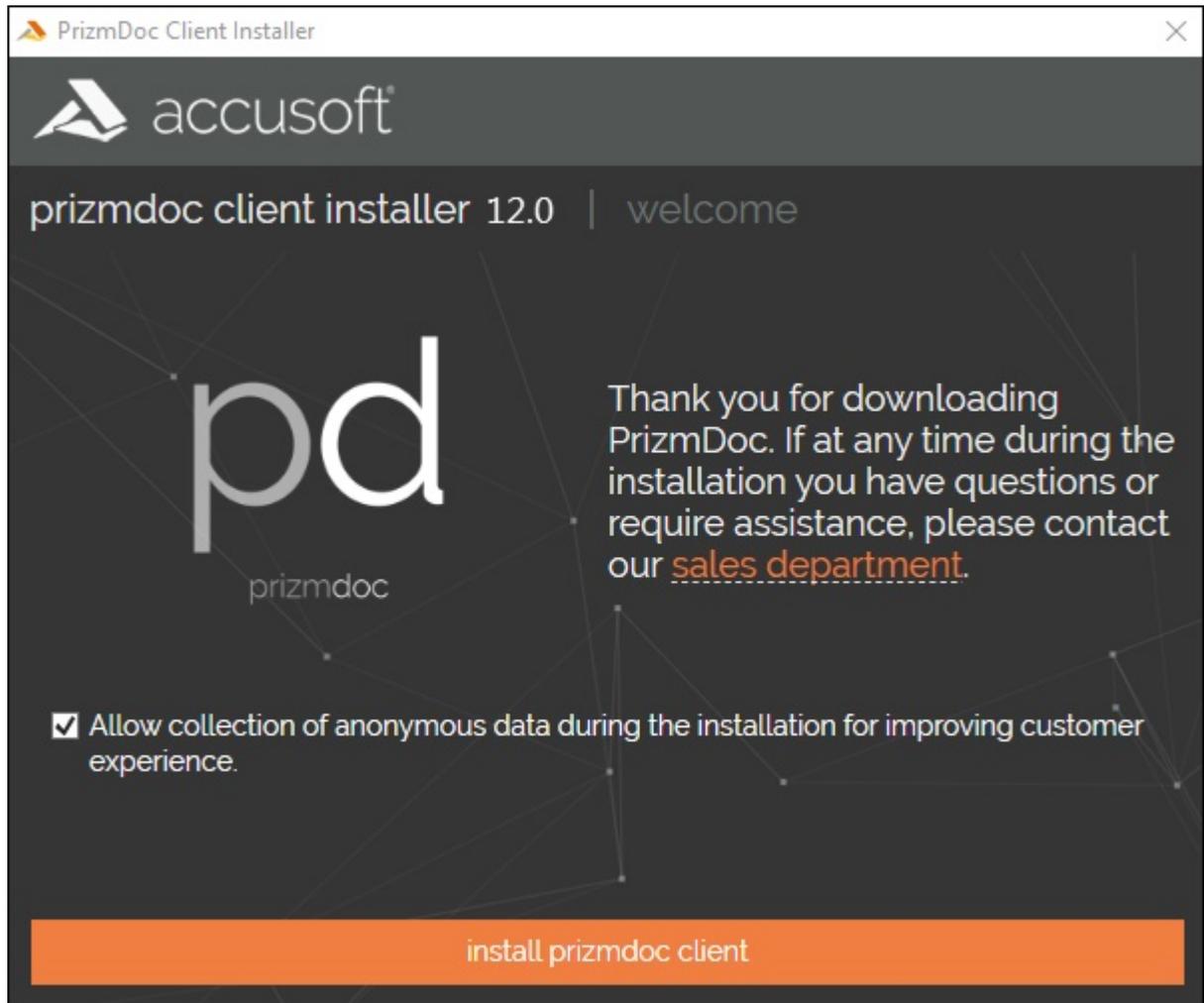
- [Install the Viewer](#)
- [Unattended Install & Uninstall](#)

## Install the Viewer

To install the PrizmDoc Viewer using an Accusoft Cloud-Hosted PrizmDoc Server, follow these steps:

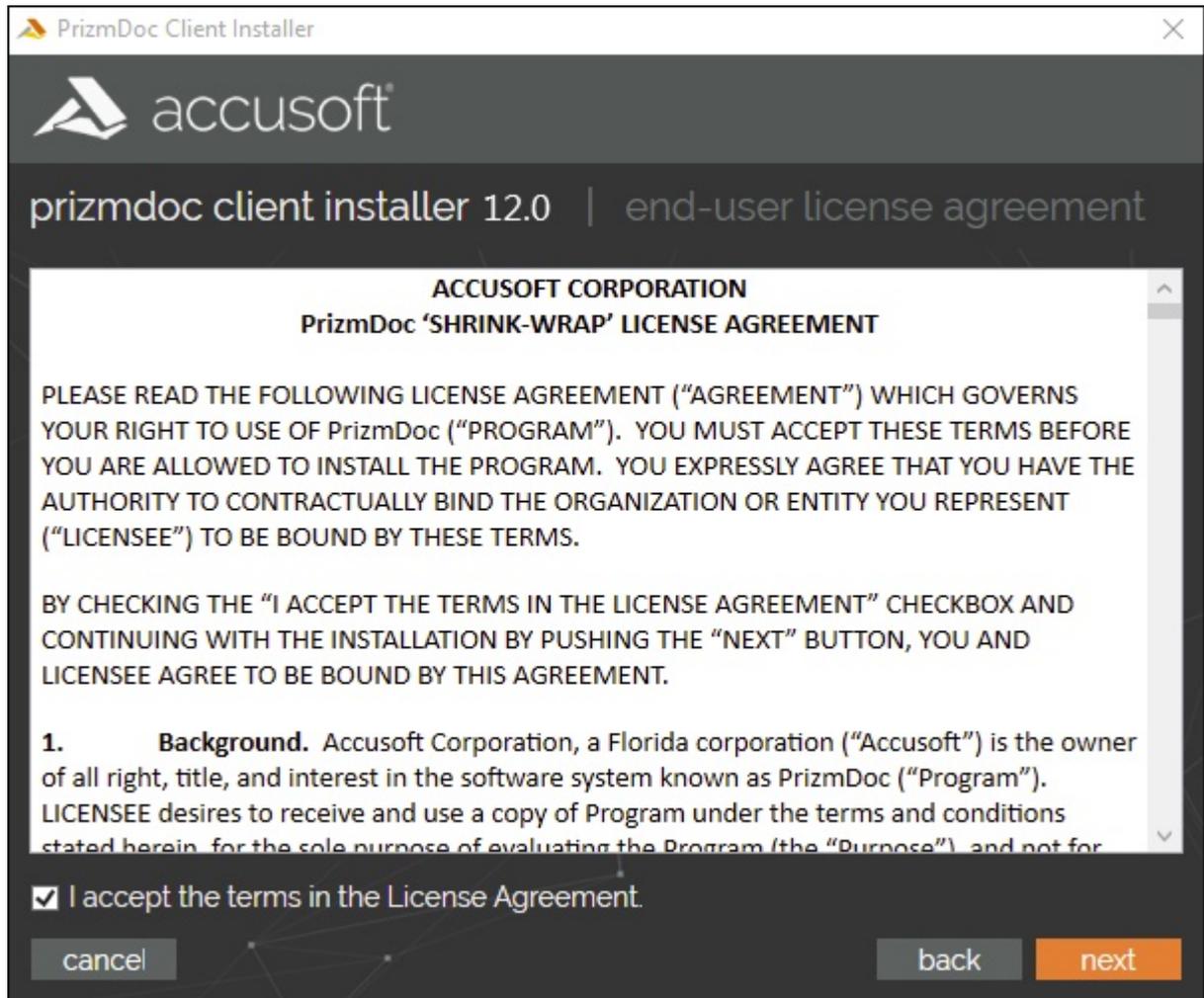
1. Start the installer from any location on your system. Click **Install PrizmDoc Client** to continue:

# PrizmDoc v12.3 - Updated June 23, 2017 97



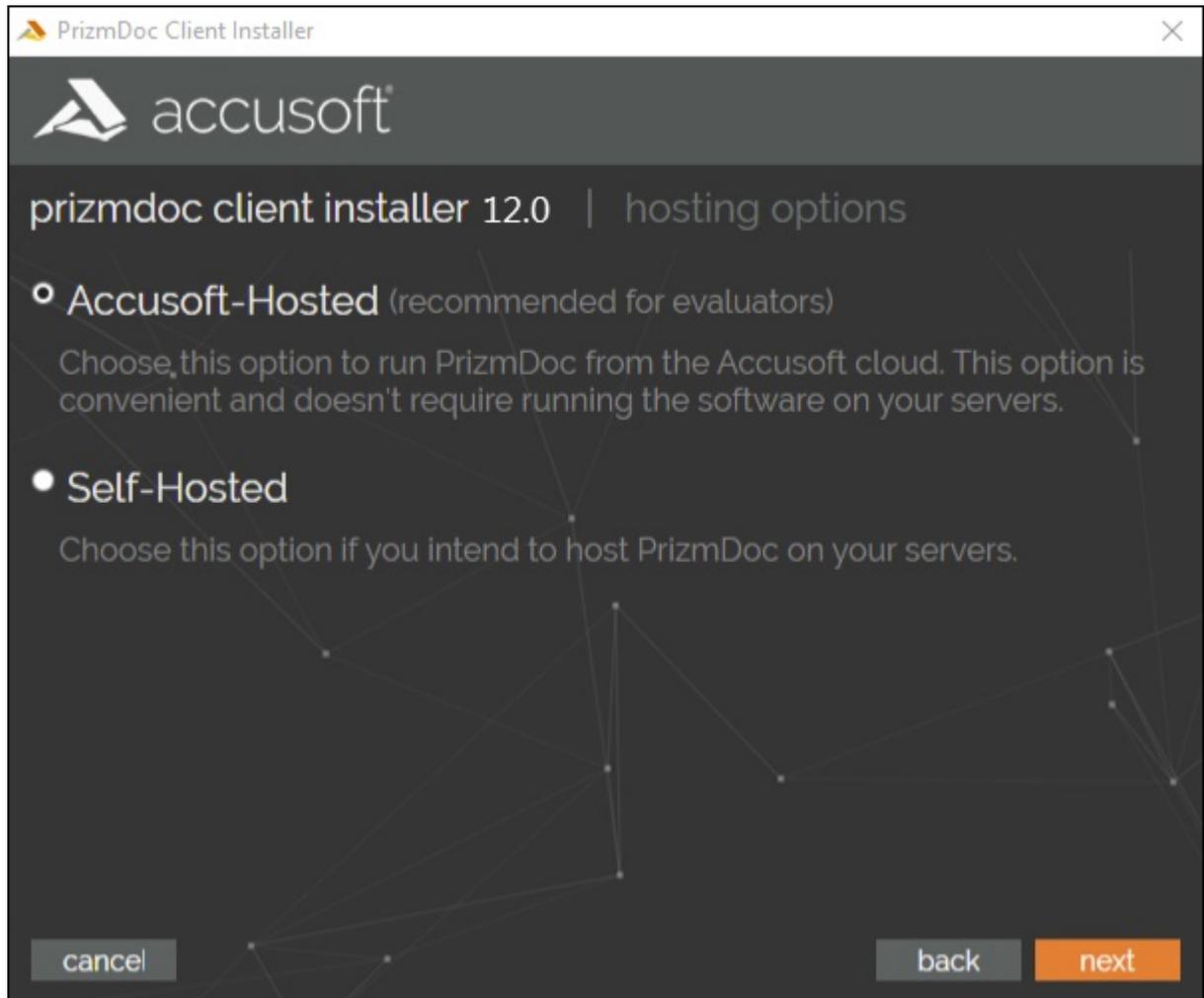
2. Carefully read the information contained in the License Agreement form before making a decision to accept the terms of the agreement. Choose **I accept the terms in the License Agreement** to accept the conditions outlined in the License Agreement and then click **Next** to continue the installation (or click **Cancel** to exit the installation process):

# PrizmDoc v12.3 - Updated June 23, 2017 98

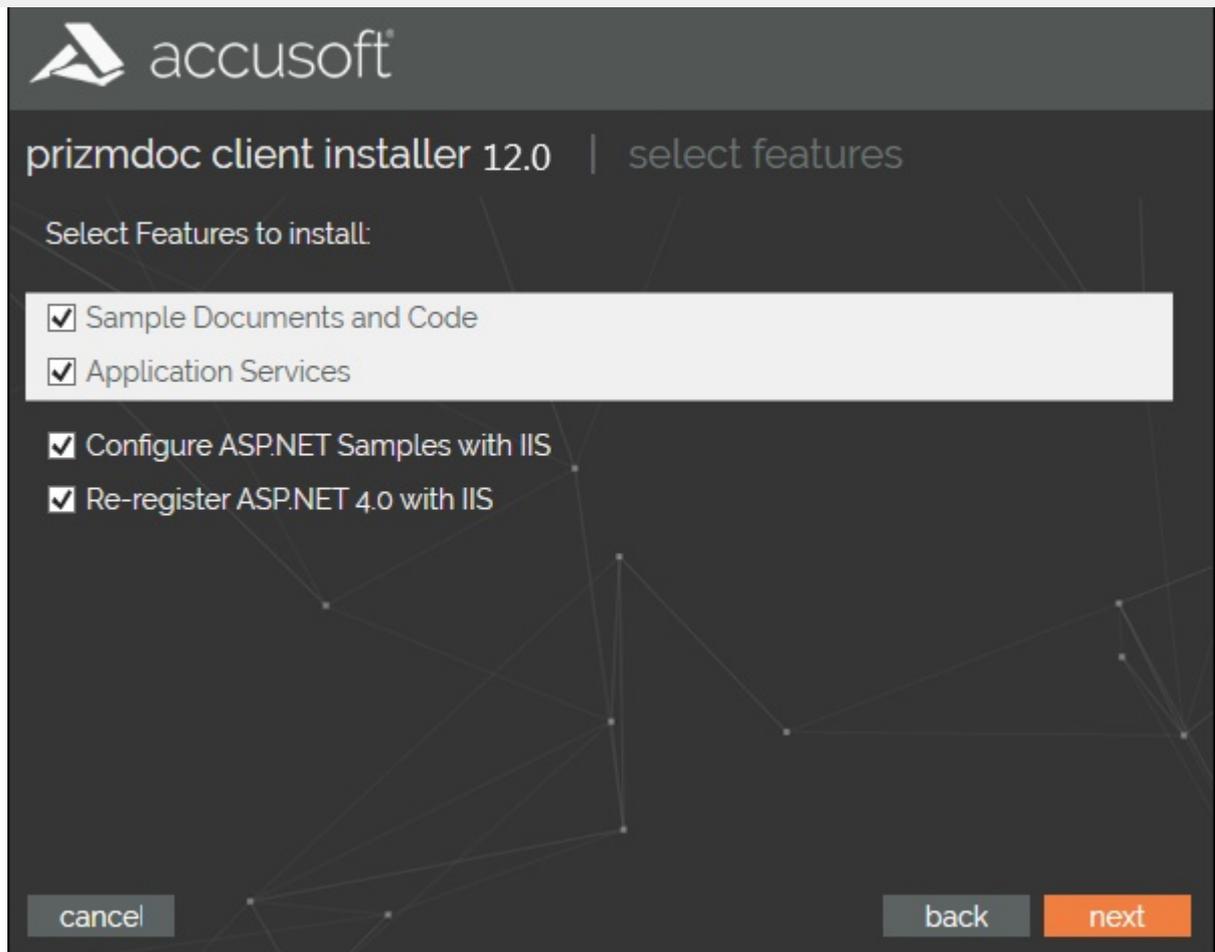


3. You have two PrizmDoc Server hosting options when installing the PrizmDoc Viewer; they are Accusoft Cloud-Hosted and Self-Hosted. For this tutorial, we are installing using an Accusoft Cloud-Hosted server. This option is highly recommended for evaluators of PrizmDoc as it does not require running the PrizmDoc Server software on your server. The Accusoft Cloud-Hosted option is selected by default. Click **Next** to continue the installation:

# PrizmDoc v12.3 - Updated June 23, 2017 99



4. The Select Features dialog is displayed:



The Select Features dialog allows you to define what components of PrizmDoc you want to install:

- **Sample Documents and Code** – Installs the different client viewers that can be used with the PrizmDoc services and sample documents.
- **Application Services** - Installs the service that provides application-level logic for the Viewer, such as enabling document viewing through the PrizmDoc Server, saving and loading of markup, and handling opening of documents and creating viewing sessions.

Additional Options:

- **Configure ASP.NET Samples with IIS.**
- **Re-register ASP.net 4.0 with IIS.**

 These options are only available if both IIS and ASP.NET 4.0+ are present.

Once you have made your selections, click **Next** to continue.

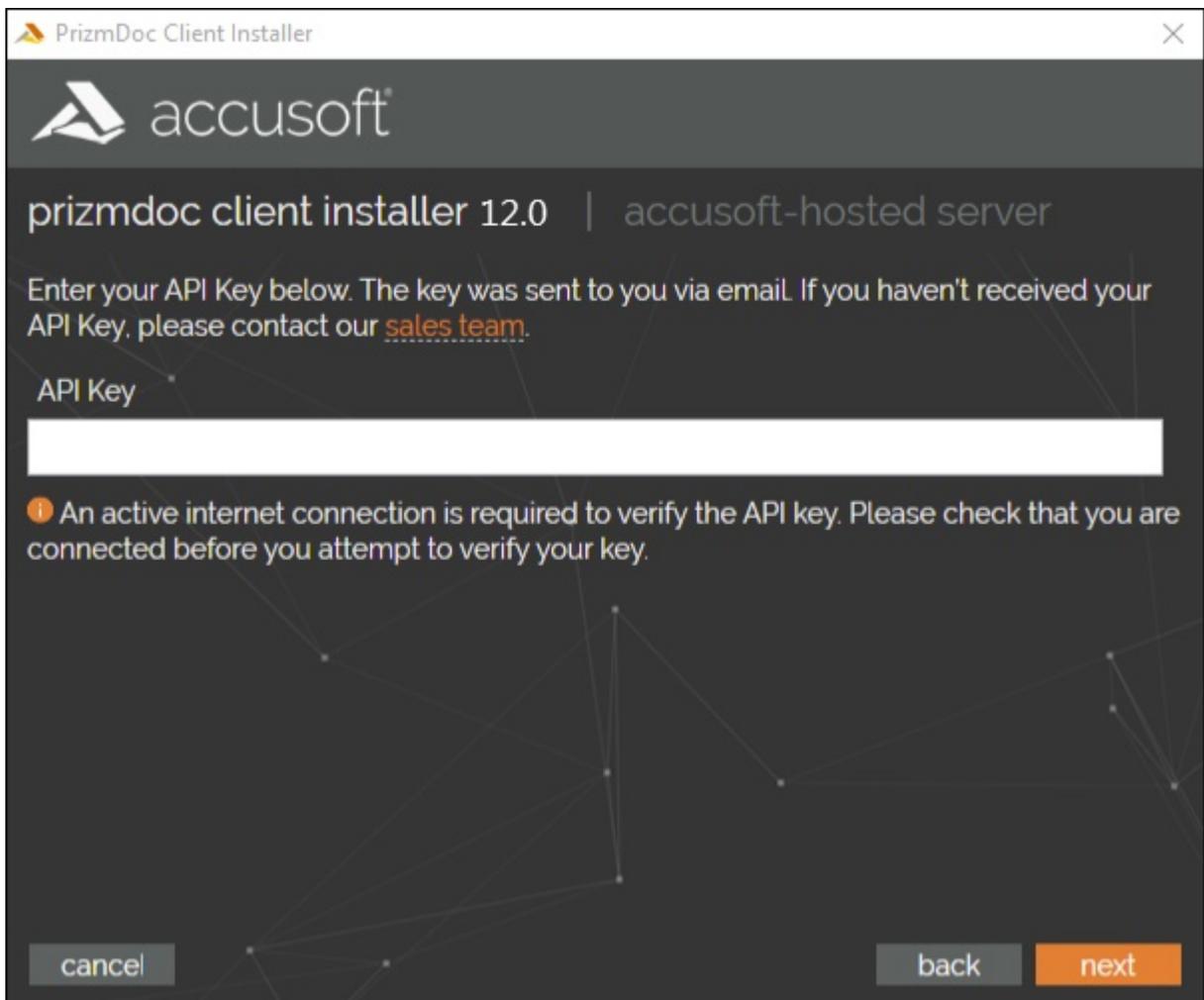
5. If you are installing the Application Services, you will need to enter your PrizmDoc API key to connect to the Accusoft Cloud-Hosted PrizmDoc Server. This key is available on the [Download page](#) and can be copied from there and pasted into API Key field below.

 You will need an active internet connection on the machine you are installing the PrizmDoc

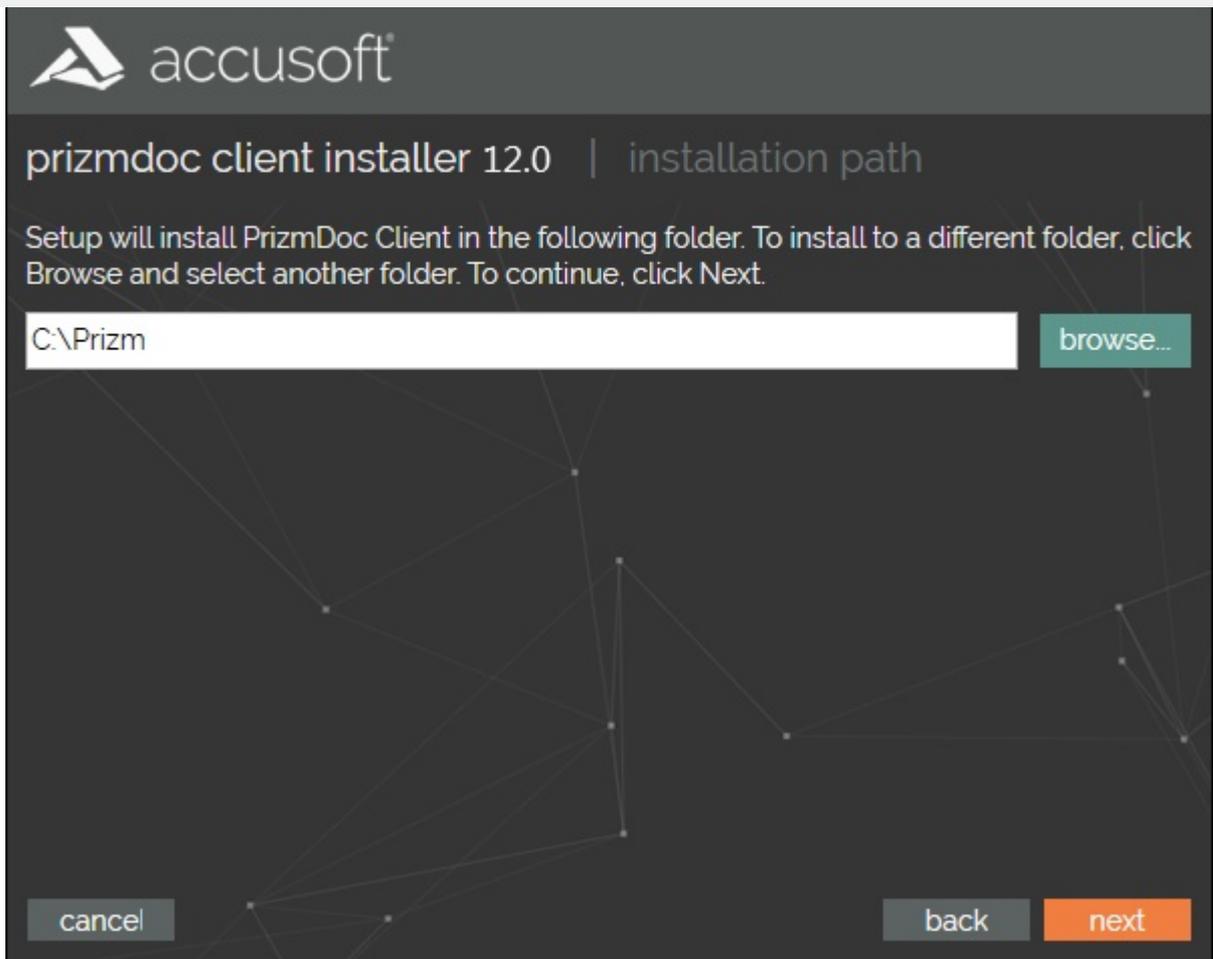
 You will not see this screen if you are not installing the Application Services at this time. Additional installation and configuration may be required:

- You may need to configure one of the Viewer samples to use an existing installation of the Application Services.
- You may need to install and configure the Application Services with your API key to connect to the Accusoft Cloud-Hosted PrizmDoc Server. See the [Application Services Configuration](#) topic for more information.

Enter your **API key** into the text field provided. Click **Next** to verify your API key and continue the installation:

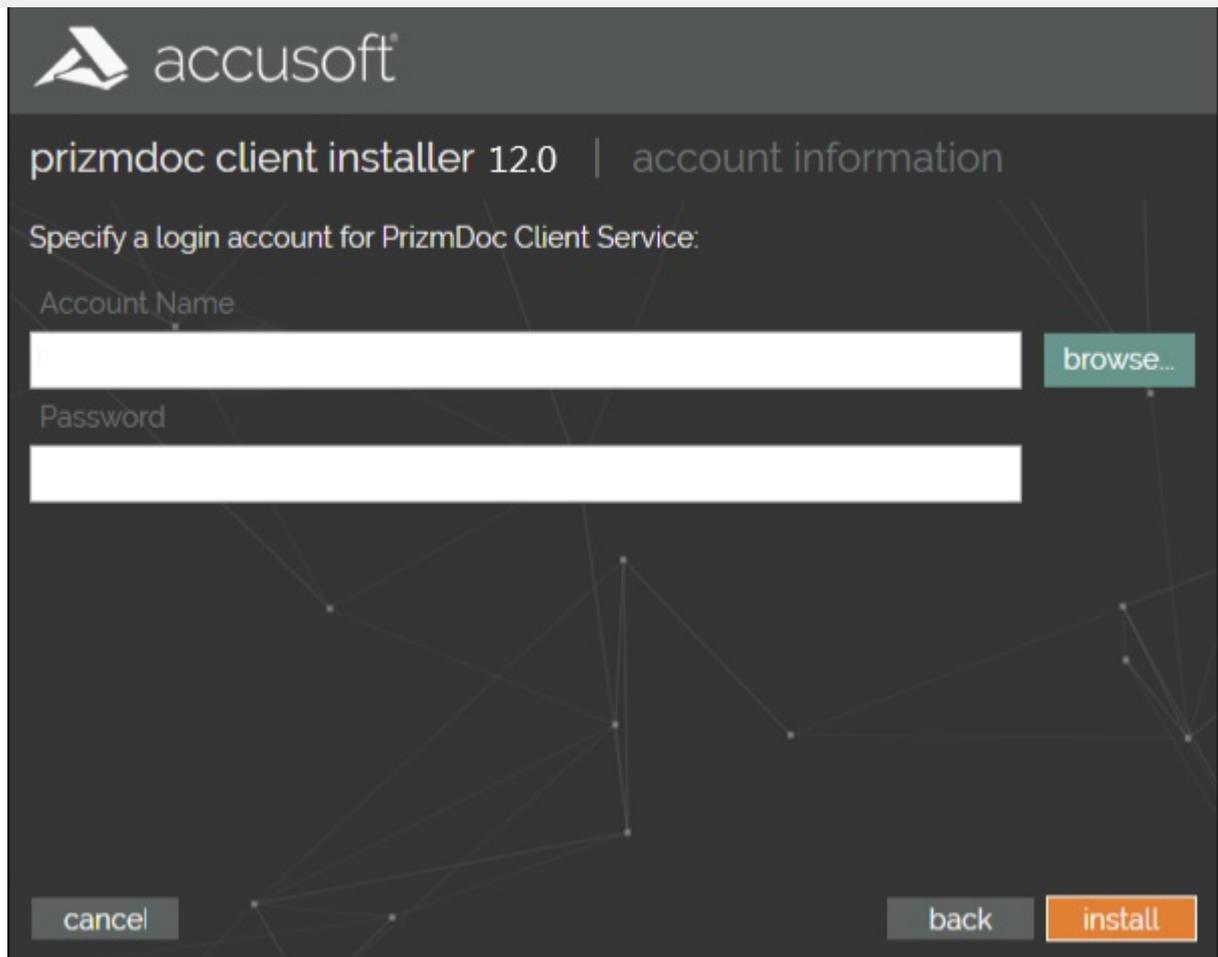


6. The Installation Path dialog is displayed. Specify the **destination directory** where the PrizmDoc Viewer should be installed, or choose the **default installation destination directory**, then click **Next**:



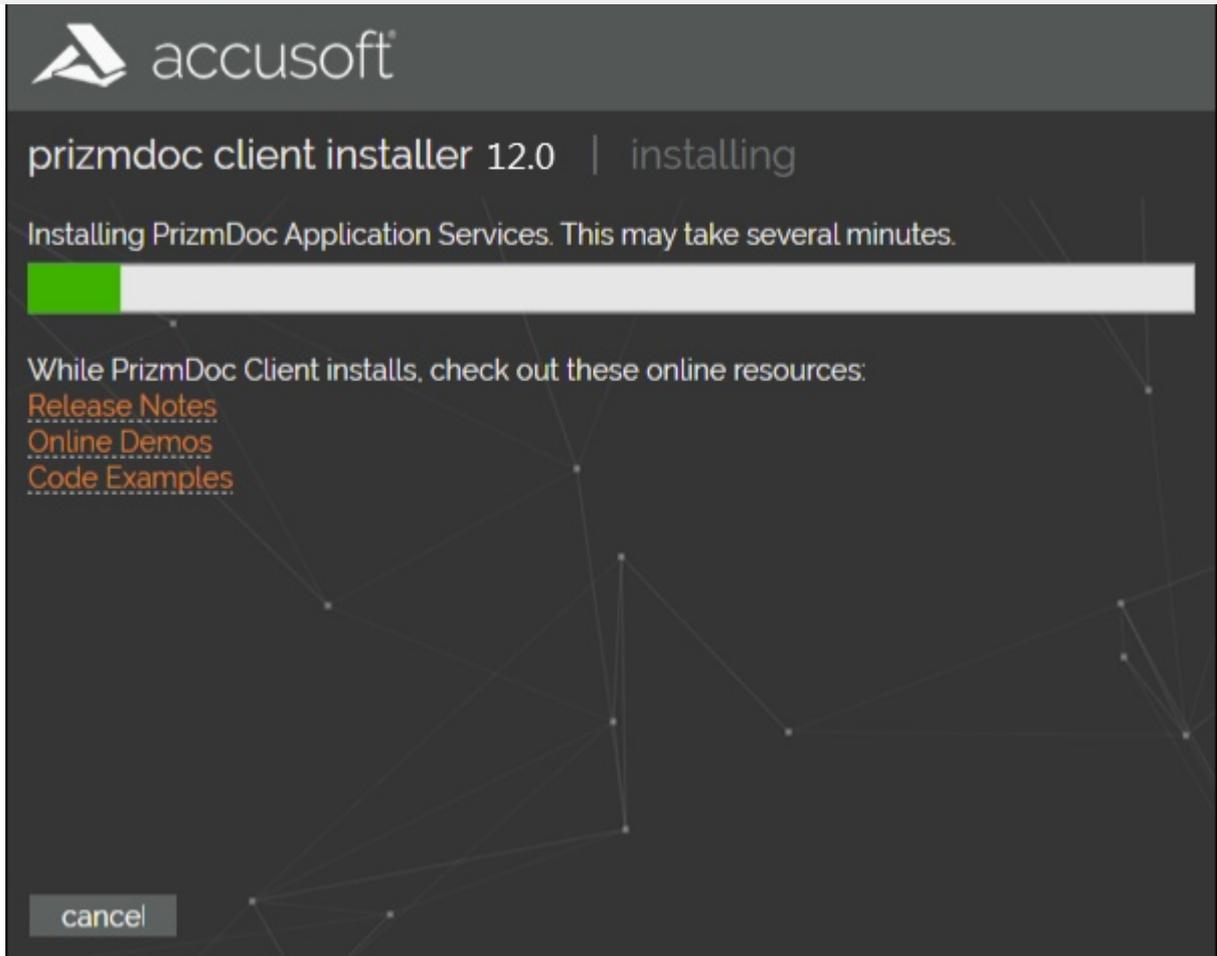
7. The next step in the installation process allows you to define the account that will be used to run the PrizmDoc Application Services that is installed on the system. The Account Information dialog will default to the user running the installation, but you can enter **a user name** or use the **Browse** button to select **a user** from the local system or domain. Once a user has been selected, enter **the password** for the user account into the dialog.

 During installation, when specifying a login account for PrizmDoc Application Services, you MUST choose a user which has administrative privileges. Otherwise, the installation will fail.



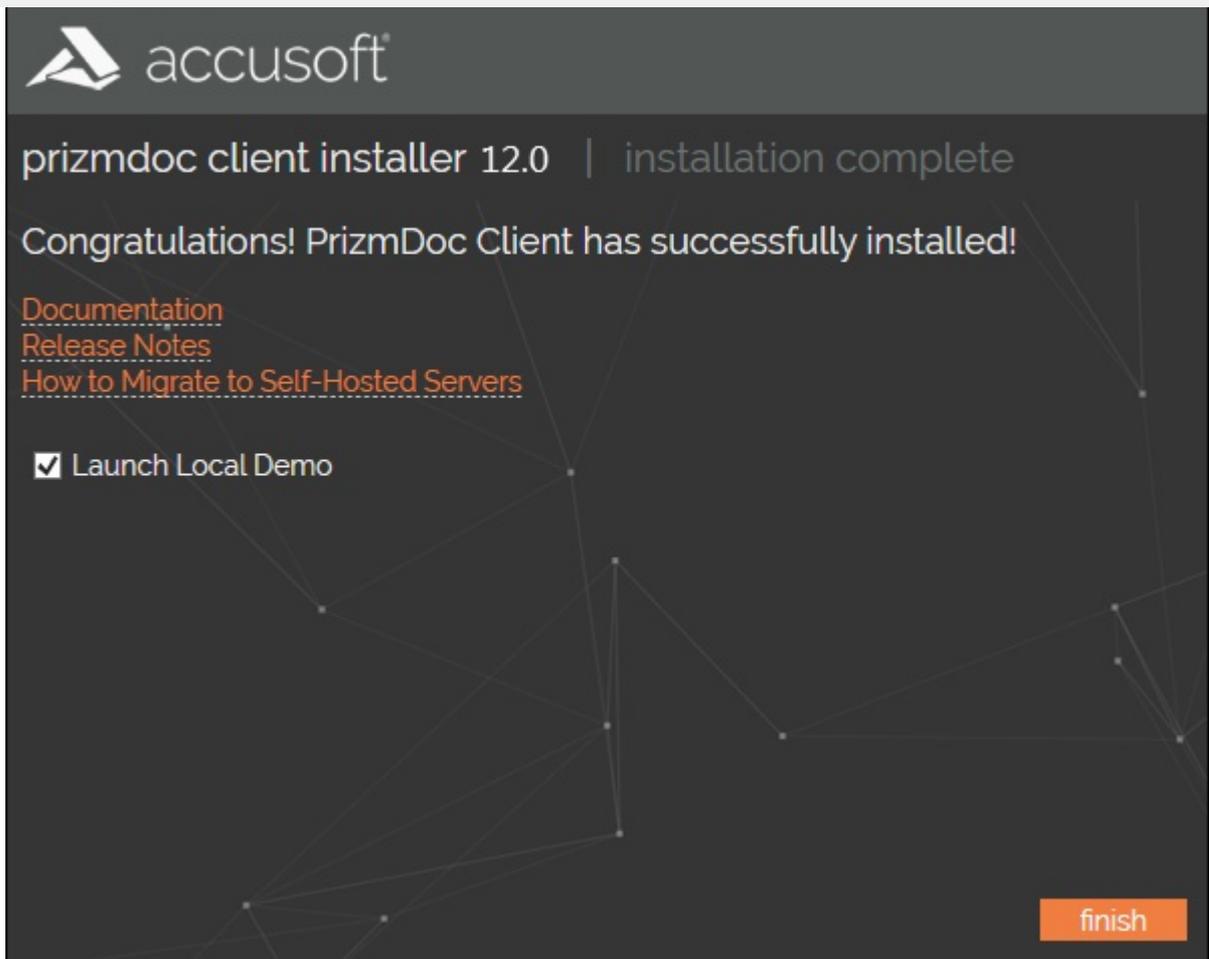
The screenshot shows a dark-themed dialog box titled "prizmdoc client installer 12.0 | account information". At the top left is the Accusoft logo. Below the title bar, the text "Specify a login account for PrizmDoc Client Service:" is displayed. There are two input fields: "Account Name" and "Password". To the right of the "Account Name" field is a "browse..." button. At the bottom of the dialog, there are three buttons: "cancel", "back", and "install". The "install" button is highlighted in orange.

8. Once you have entered in all of the appropriate information, click **Install** to continue. (If the password is not correct an error dialog will be displayed noting that the password is not correct.) The Installation dialog is displayed with a progress bar:



The Installer will unpack the product, and lay out and configure the product as defined in the configuration dialogs. While PrizmDoc is installing, you can look at the **Release Notes**, **Online Demos**, or **Code Examples** from the links provided.

9. Once the installation is complete, the Installation Complete dialog is displayed:



There are links to the Documentation, product Release Notes, as well as a How to Migrate topic if you wish to view them directly after the installation. There will also be an option to launch the locally installed demo in your default browser.

The option to launch the local demo is only available if all of the following occurred:

- Sample Documents and Code were installed.
- The Samples were configured and re-registered with IIS.
- The PrizmDoc Application Services were installed.

10. Click **Finish** to exit the Installer.

11. Go to [Check the Connection to Accusoft Cloud-Hosted Services](#) for instructions on verifying that your installation of the PrizmDoc Viewer is able to contact the back-end services.

## Unattended Install & Uninstall

### Unattended Install

The PrizmDoc Viewer Windows installer can be installed unattended, however certain properties must be set:

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.	None
ServicePassword	Required - The password for the ServiceUser.	None
InstallFolder	Optional - The base installation directory for the product.	"C:\Prizm"
SelectedClientFeatures	Optional - The Viewer features to install. This can be empty or a comma separated list of values that can contain any of the following: <ul style="list-style-type: none"> <li>"HTML5Viewer"</li> <li>"LocalFileViewerFeature"</li> </ul>	"HTML5Viewer, LocalFileViewerFeature"
IISConfigureSamples	Optional - Whether to configure the samples with IIS or not. Set to "1" for yes and set to an empty string for no.	"1"
IISReregister	Optional - Whether to re-register ASP.NET v4 with IIS or not. Set to "1" for yes and set to an empty string for no.	"1"
SelectedPASFeatures	Optional - This can be set to "ALL" to include the PrizmDoc Application Services (PAS) features or set to an empty string to not include them.	"ALL"
ApiKey	Optional - The API key required to use the Accusoft Cloud-Hosted PrizmDoc servers.  Note: You will not be able to make requests against the Accusoft Cloud-Hosted servers without a valid API key.	None

To start the unattended install:

1. The PrizmDocClient.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, ServicePassword, and ApiKey are examples, and you will need to change them to specify your ServiceUser, ServicePassword, and ApiKey):

```
> PrizmDocClient.exe ServiceUser=accusoft.com\PrizmUser
ServicePassword=pdpassword ApiKey=AccusoftHostedKey -s -l output.log
```

 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 4

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.

```
> start /wait PrizmDocClient.exe ServiceUser=accusoft.com\PrizmUser  
ServicePassword=pdpassword -s -l output.log
```

4. PrizmDoc should now be installed, licensed and started.

## Unattended Uninstall

You can use the -u flag as shown below to silently uninstall PrizmDoc on Windows:

```
> PrizmDocClient.exe -s -u -l output.log
```

## Install on Linux

To install the PrizmDoc Viewer using an Accusoft Cloud-Hosted PrizmDoc Server, follow the steps provided in this section:

Some steps are specific to a particular Linux distribution; these steps will be labeled as being specific to one of the following:

- Red Hat / Fedora (CentOS) Linux Distributions
- Debian (Ubuntu) Linux Distributions

 Make sure you log in as **root** to the machine.

### Step 1 - Download the PrizmDoc Viewer Linux Distribution

 Packages are only available for 64-bit systems.

Before downloading PrizmDoc, you will need to purchase a license key or request a Trial Evaluation by filling out the appropriate form at [www.accusoft.com](http://www.accusoft.com).

Once you have filled out the form for a Trial Evaluation, you can download PrizmDoc by doing one of the following:

1. Following the link provided in the response email and selecting the desired Linux distribution.

OR

2. Downloading directly to the Linux server using the 'wget' command for the specific distribution as shown below:

 **You must substitute the version of the package you are using in the code examples below.** For example, if you are using v11.0, then specify "11.0" instead of "<version>". If the version is a hot fix, you will need to also specify the hot fix number, for example, "11.0.1".

### Red Hat, Fedora, CentOS, and Older Linux Distributions

#### Example

```
wget
```

## Debian and Ubuntu Linux Distributions

### Example

```
wget
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_client_<version>.amd64.deb.tar.gz
```

## Generic .tar.gz Distribution

### Example

```
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, Fedora, CentOS, and Older Linux Distributions

The following example is for Red Hat, Fedora, CentOS, and older Linux distributions:

### Example

```
tar -xzvf prizmdoc_client_*.rpm.tar.gz
cd prizmdoc_client_*.rpm
yum install --nogpgcheck *.rpm
```

## Debian (Ubuntu) Linux Distributions

The following example is for Debian (Ubuntu) Linux distributions:

### 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 command
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

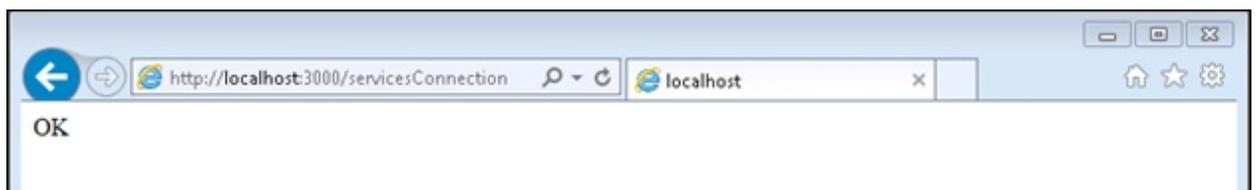
```
tar -xzvf prizmdoc_client_*.tar.gz
cd prizmdoc_client_*
ls prizm-*.tar.gz | xargs -n1 tar zxf
cp -R prizm /usr/share/
```

1. After installation, locate the PrizmDoc Application Services config file. Assuming the standard install location, this is **/usr/share/prizm/pas/pcc.nix.yml**.
2. Edit the file and specify the following values:
  - **pccServer.hostName**: "api.accusoft.com"
  - **pccServer.port**: 443
  - **pccServer.scheme**: "https"
3. Set **pccServer.apiKey** to your API key. This key will have been emailed to you as part of the registration process for downloading the installer. For more information on your PrizmDoc API key, refer to the topic, [How to Get an Evaluation License](#).
4. Restart **PrizmDoc Application Services** for the changes to take effect. See [Starting & Stopping Application Services](#) for instructions.
5. Go to [Check Your Connection to Accusoft Cloud-Hosted Services](#) for instructions on verifying that your installation of the PrizmDoc Viewer is able to contact the back-end services.

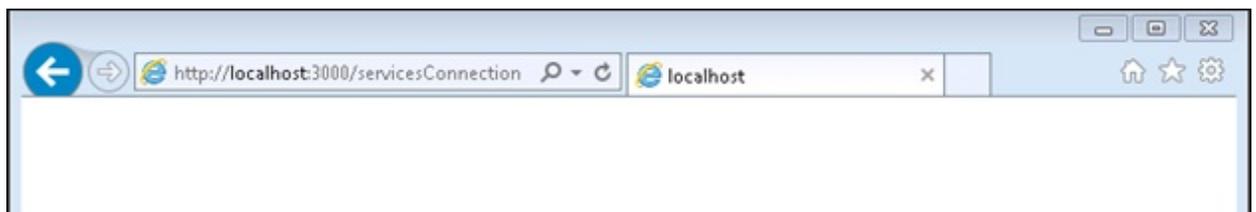
## Check the Connection to Accusoft Cloud-Hosted Services

To verify that your installation of PrizmDoc is able to contact the back-end services, you can utilize PrizmDoc Application Services (PAS) to check your connection to PrizmDoc Server hosted by Accusoft.

1. On the machine where you installed PrizmDoc Application Services, 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 PrizmDoc Application Services.
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:



 For more information, refer to the [Troubleshooting](#) section.

3. Once you are done checking the connection, go to [Step 3 - Integrate the Viewer with Your Application](#).

## Self-Hosted

This section contains the following information:

- [System Requirements & Supported Environments](#)
- [Install PrizmDoc Viewer](#)
  - [Install on Windows](#)
    - [Install the Viewer](#)
    - [Unattended Install & Uninstall](#)
  - [Install on Linux](#)
    - [Linux Installation](#)
    - [Configure Server Connections](#)
    - [Uninstall PrizmDoc on Linux](#)
- [Check the Connection to your Self-Hosted Server](#)

## System Requirements & Supported Environments

This section provides information about the system requirements for PrizmDoc Application Services (PAS) when using a Self-Hosted installation:

 PrizmDoc is only supported on 64-bit operating systems.

### PrizmDoc Application Services

PrizmDoc Application Services is a suite of RESTful APIs that is primarily a proxy for PrizmDoc Server. It requires significant network throughput, disk I/O, and very little processing power.

### Supported Operating Systems

#### Windows

 Product test validation on the following Operating Systems are deprecated in v12.2 and will be discontinued altogether in v13.0:

- Windows 7
  - Windows 8
  - Windows 10
- Windows Server 2008 R2 SP1
  - Windows Server 2012
  - Windows Server 2012 R2

 Product test validation on the following Operating Systems are deprecated in v12.2 and will be discontinued altogether in v13.0:

- CentOS 5.9
- Red Hat Enterprise Linux 5.9
- Ubuntu 12.04 LTS, 13.04

- CentOS 6.4, 7
- Debian 7.1 (Wheezy)
- Red Hat Enterprise Linux 6.4, 7
- Ubuntu 14.04 LTS

## 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 Application Services uses and how to optimize them for your needs.

## Install PrizmDoc Viewer

This section contains the following information:

- [Install on Windows](#)
  - [Install the Viewer](#)
  - [Unattended Install & Uninstall](#)
- [Install on Linux](#)
  - [Linux Installation](#)
  - [Configure Server Connections](#)
  - [Uninstall PrizmDoc on Linux](#)

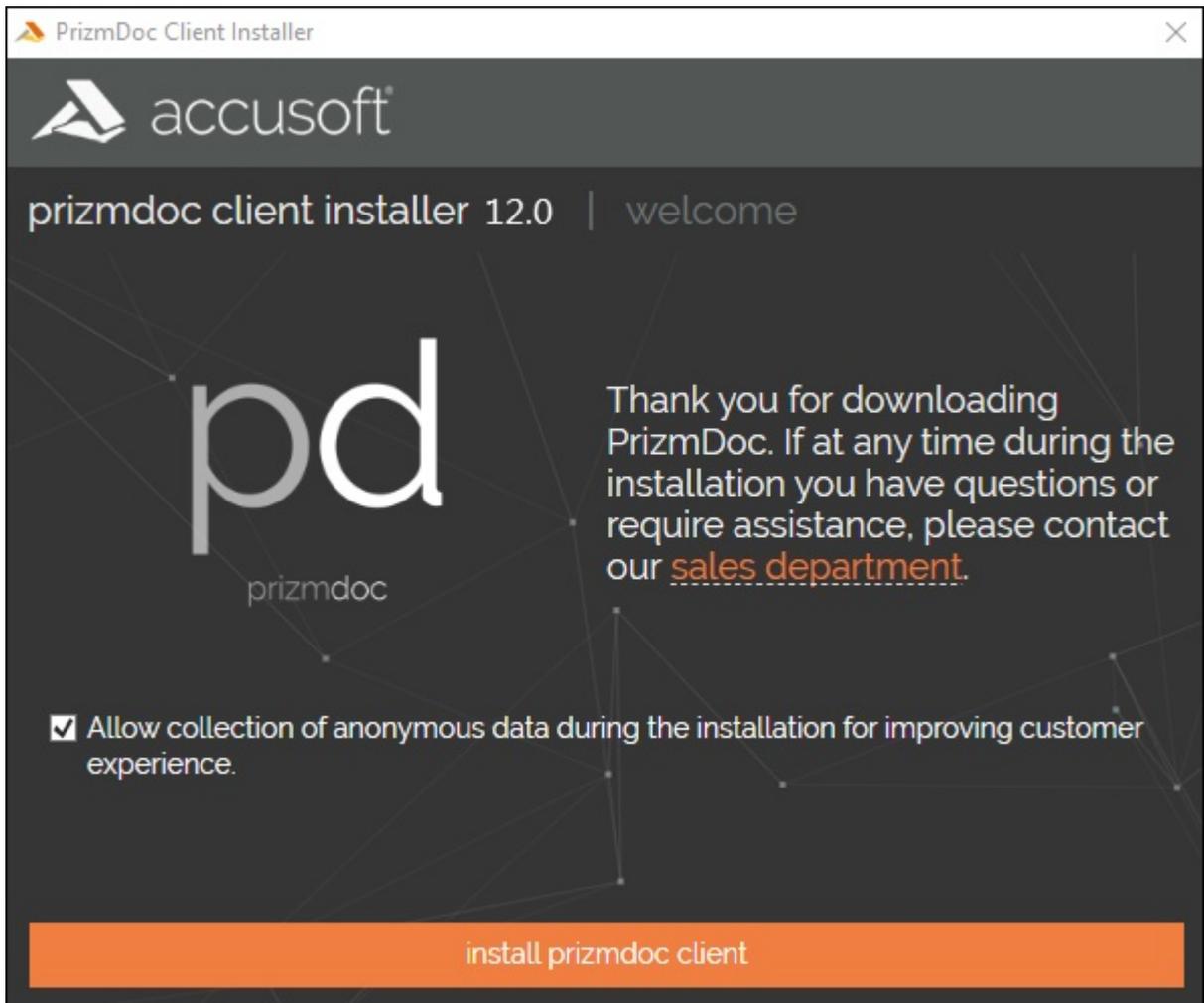
## Install on Windows

This section contains the following information:

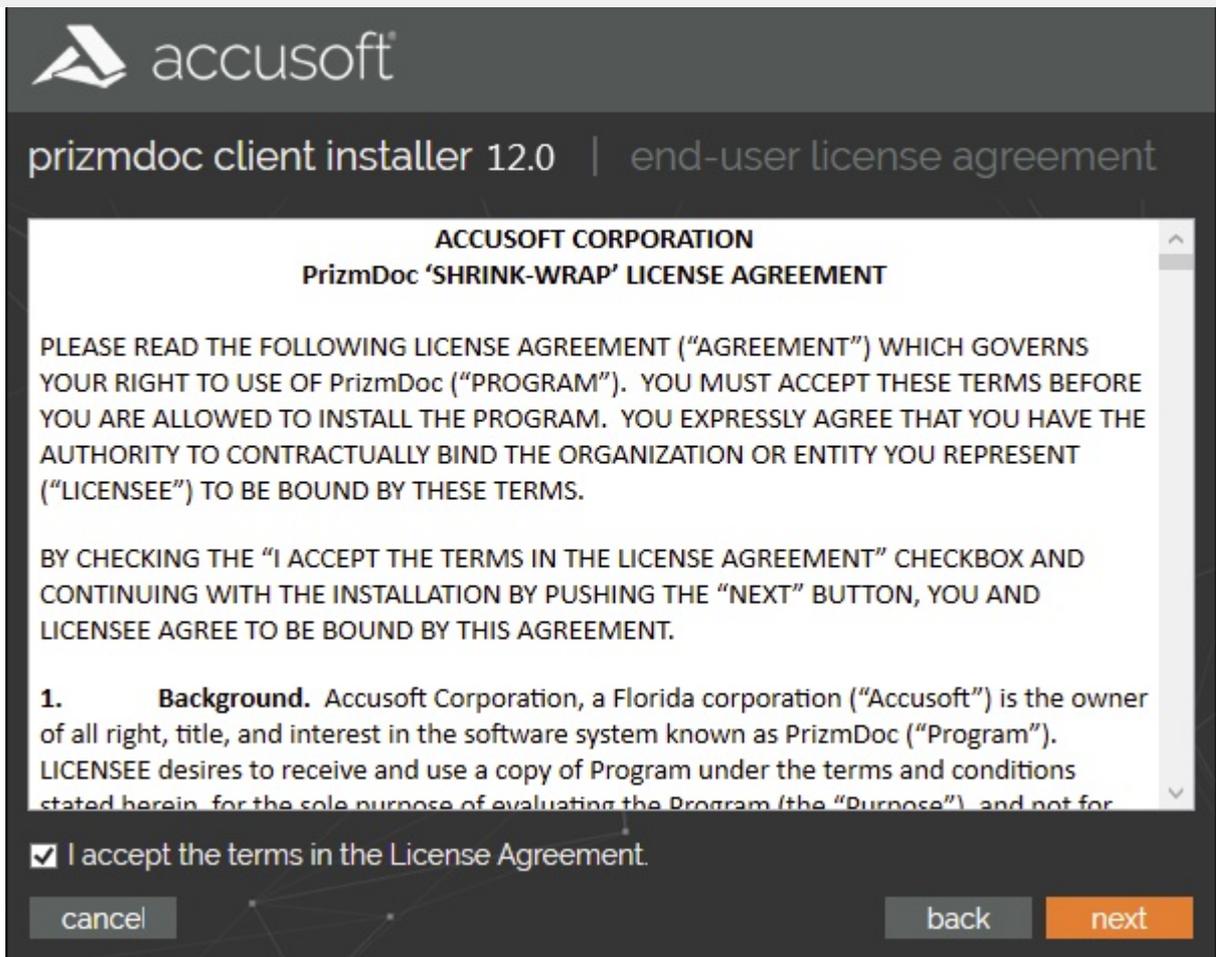
- [Install the Viewer](#)
- [Unattended Install & Uninstall](#)

## Install the Viewer

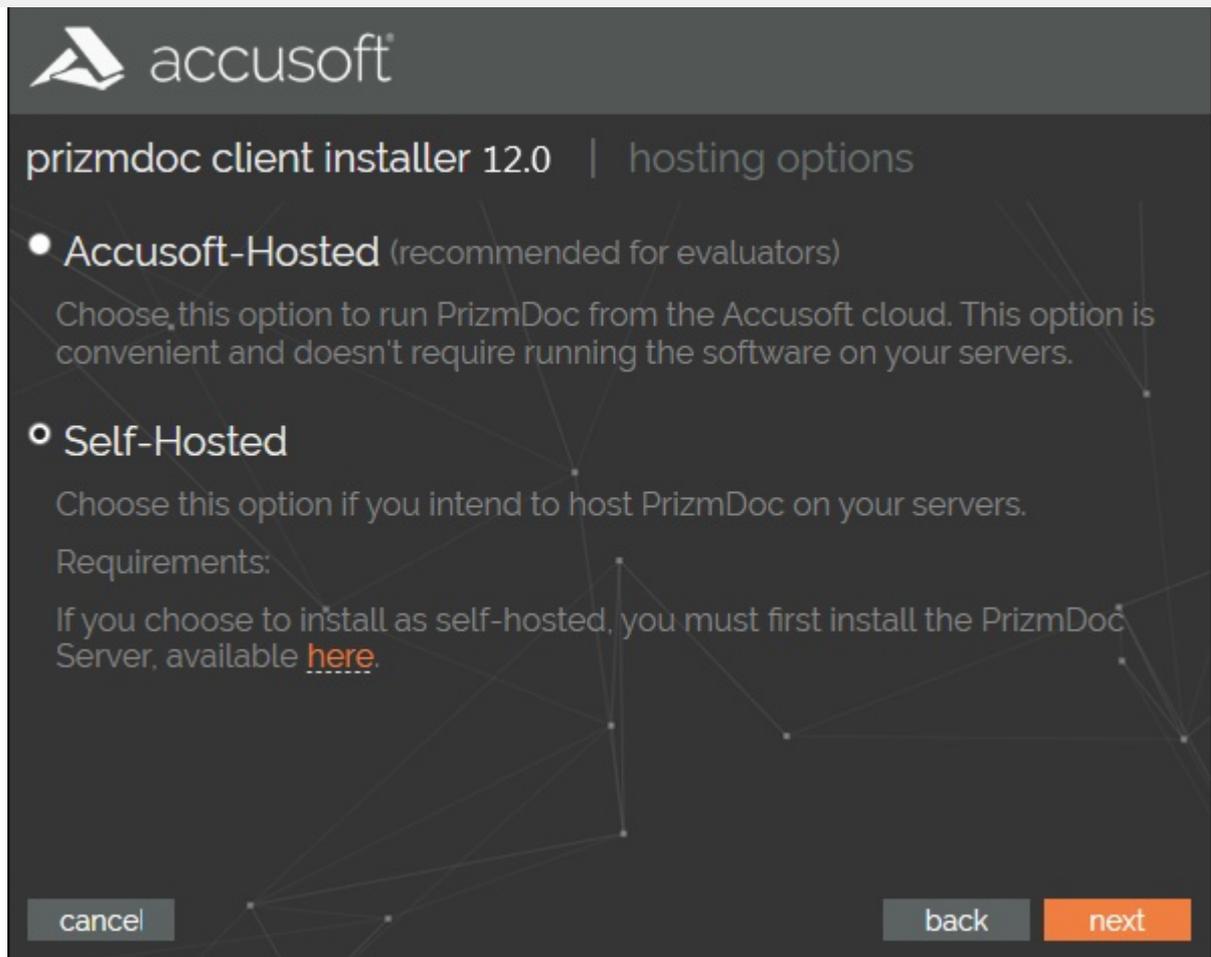
To install the PrizmDoc Viewer using a Self-hosted PrizmDoc Server, follow these steps:



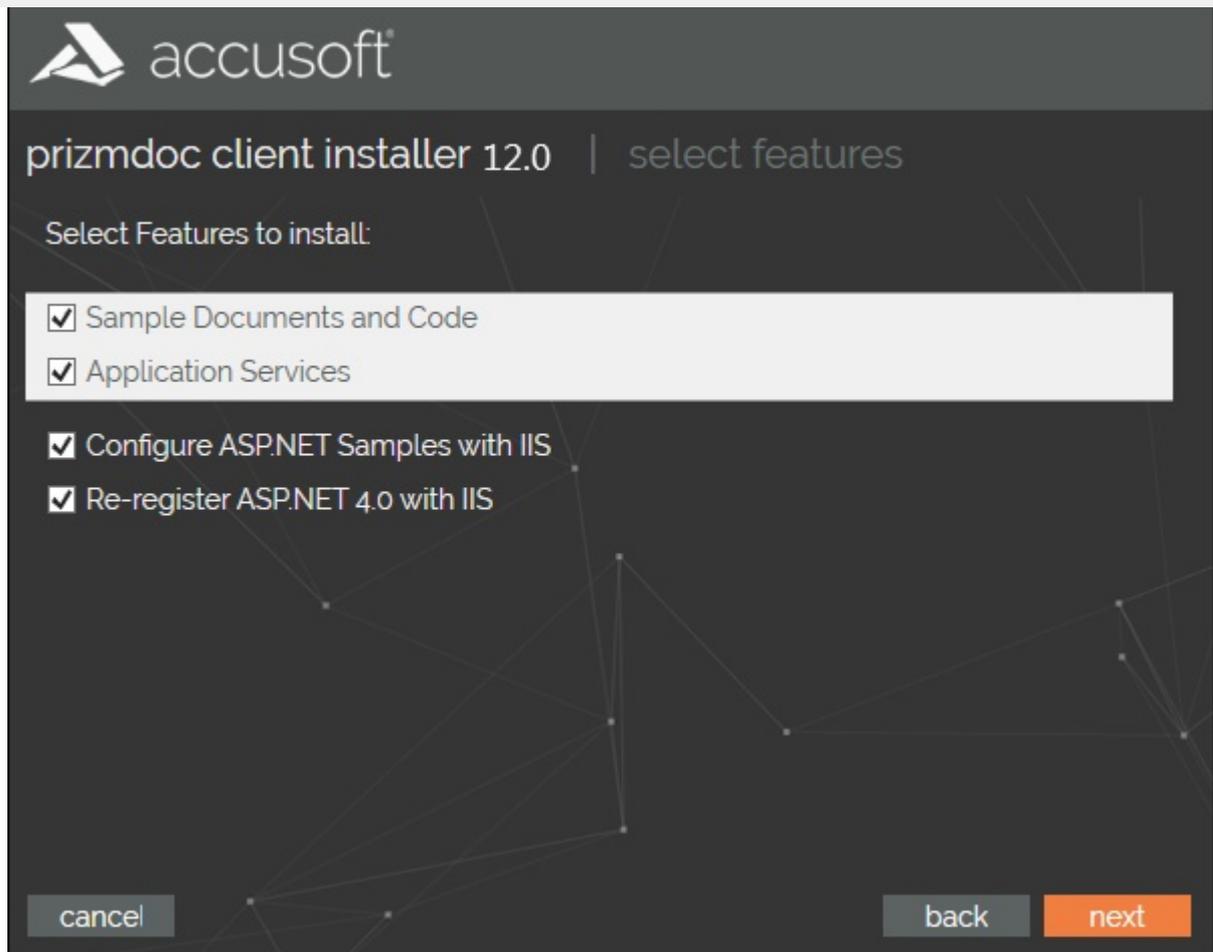
2. Carefully read the information contained in the License Agreement form before making a decision to accept the terms of the agreement. Choose **I accept the terms in the License Agreement** to accept the conditions outlined in the License Agreement and then click **Next** to continue the installation (or click **Cancel** to exit the installation process):



3. You have two PrizmDoc Server hosting options when installing the PrizmDoc Viewer; they are Accusoft Cloud-Hosted and Self-Hosted. For this tutorial, we are installing using a Self-Hosted server. Click the **Self-Hosted** option. Click **Next** to continue the installation:



4. The Select Features dialog is displayed:



The Select Features dialog allows you to define what components of PrizmDoc you want to install:

- **Sample Documents and Code** – Installs the different client viewers that can be used with the PrizmDoc services and sample documents.
- **Application Services** - Installs the service that provides application-level logic for the Viewer, such as enabling document viewing through the PrizmDoc Server, saving and loading of markup, and handling opening of documents and creating viewing sessions.

Additional Options:

- **Configure ASP.NET Samples with IIS.**
- **Re-register ASP.net 4.0 with IIS.**

 These options are 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 Application Services, you have the option to enter the PrizmDoc **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.

 You will need an active internet connection on the machine you are installing the PrizmDoc Viewer to test connectivity to your self-hosted PrizmDoc Server.

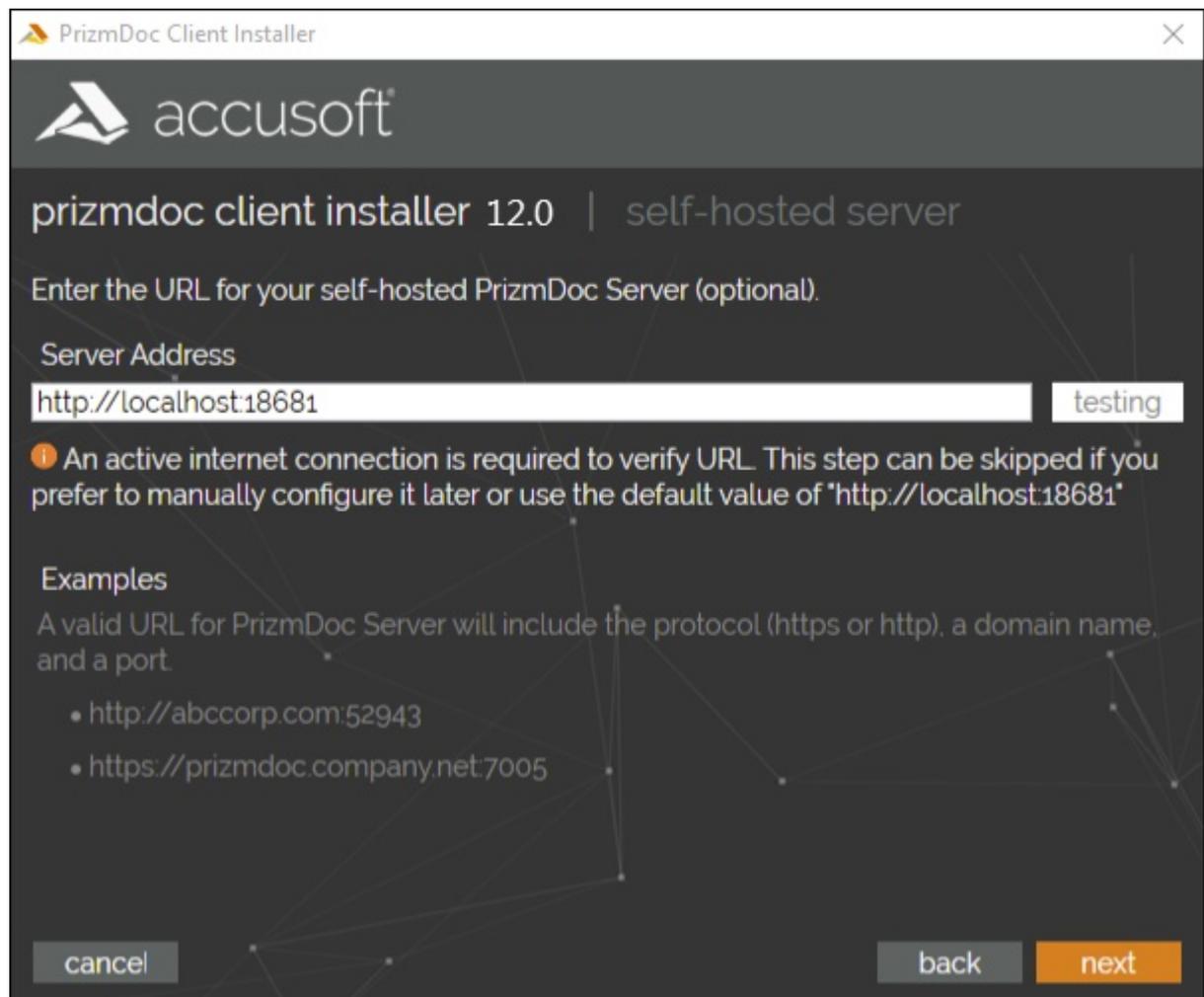
access to your PrizmDoc Server.

 A valid URL for PrizmDoc server will include an http or https, a domain name, and a port number.

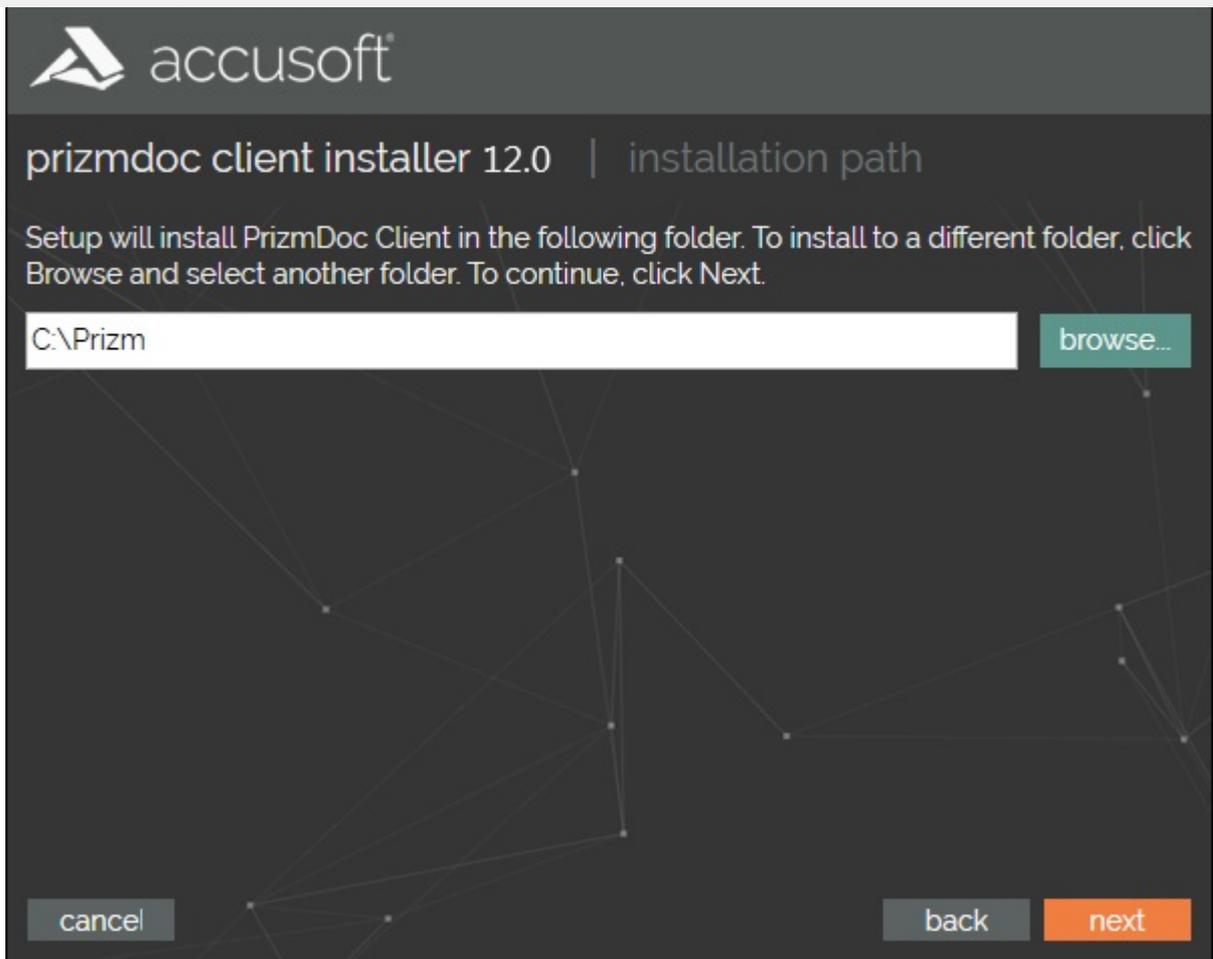
 You will not see this screen if you are not installing the Application Services at this time. Additional installation and configuration may be required:

- You may need to configure one of the Viewer samples to use an existing installation of the Application Services.
- You may need to install and configure the Application Services with your PrizmDoc Server URL to connect to the Self-hosted PrizmDoc Server. See the [Application Services Configuration](#) topic for more information.

Click **Next** to continue the installation.

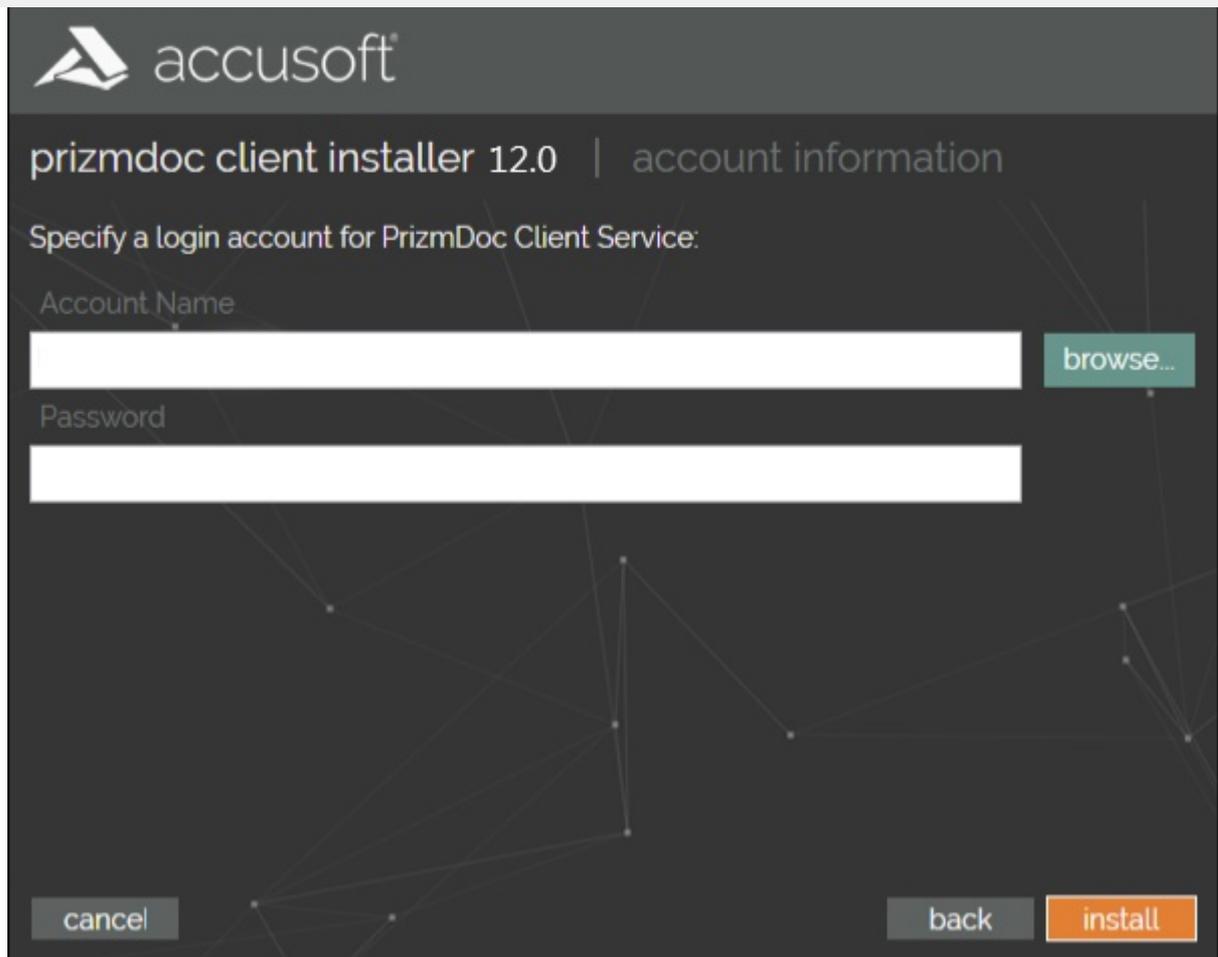


6. The Installation Path dialog is displayed. Specify the **destination directory** where the PrizmDoc Viewer should be installed, or choose the **default installation destination directory**, then click **Next** to continue:



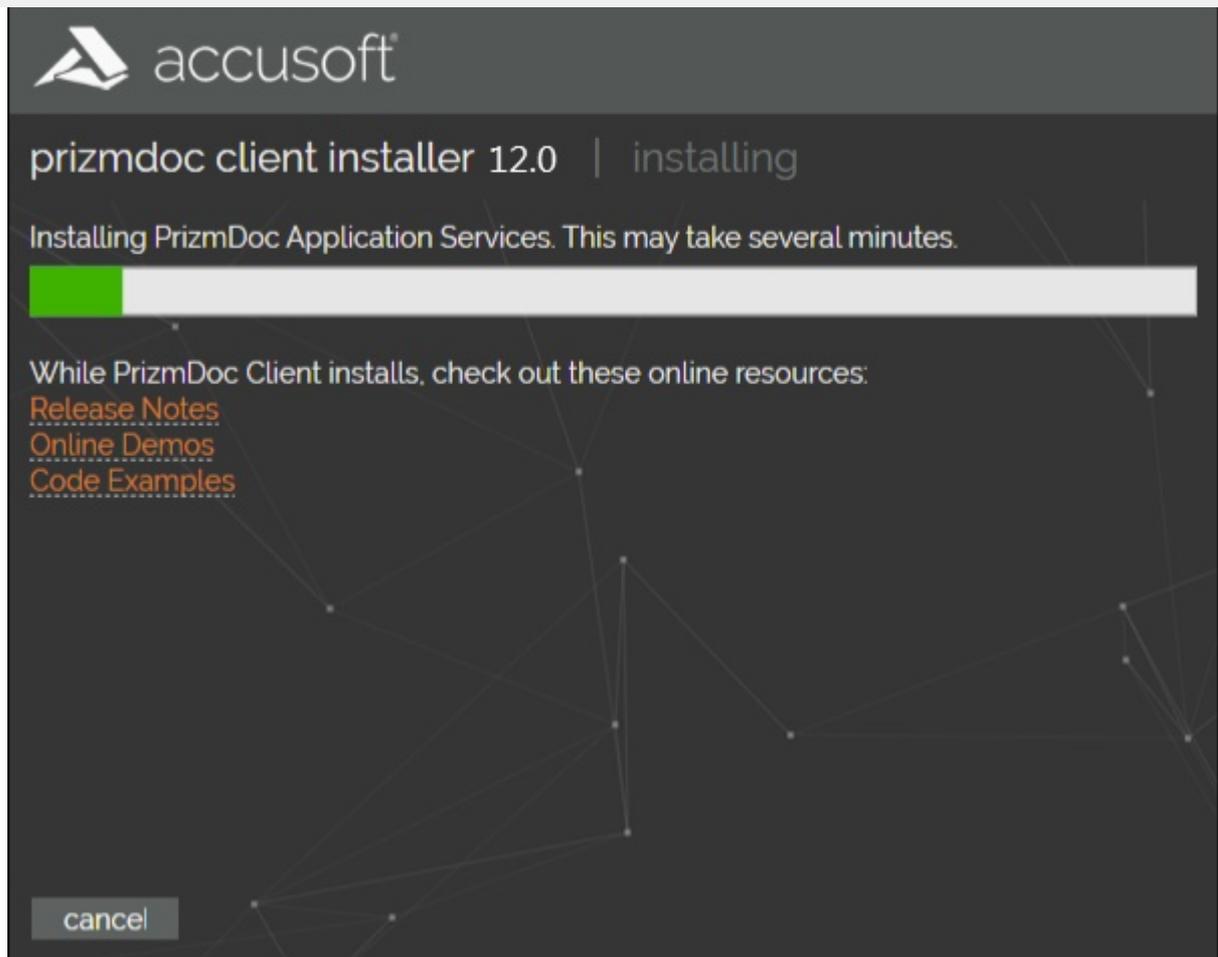
7. The next step in the installation process allows you to define the account that will be used to run the PrizmDoc Application Services that is installed on the system. The Account Information dialog will default to the user running the installation, but you can enter in a **user name** or use the **Browse button** to **select a user from the local system or domain**. Once a user has been selected, the password for the user account will need to be entered into the dialog.

 During installation, when specifying a login account for PrizmDoc Application Services, you **MUST** choose a user which has administrative privileges. Otherwise, the installation will fail.



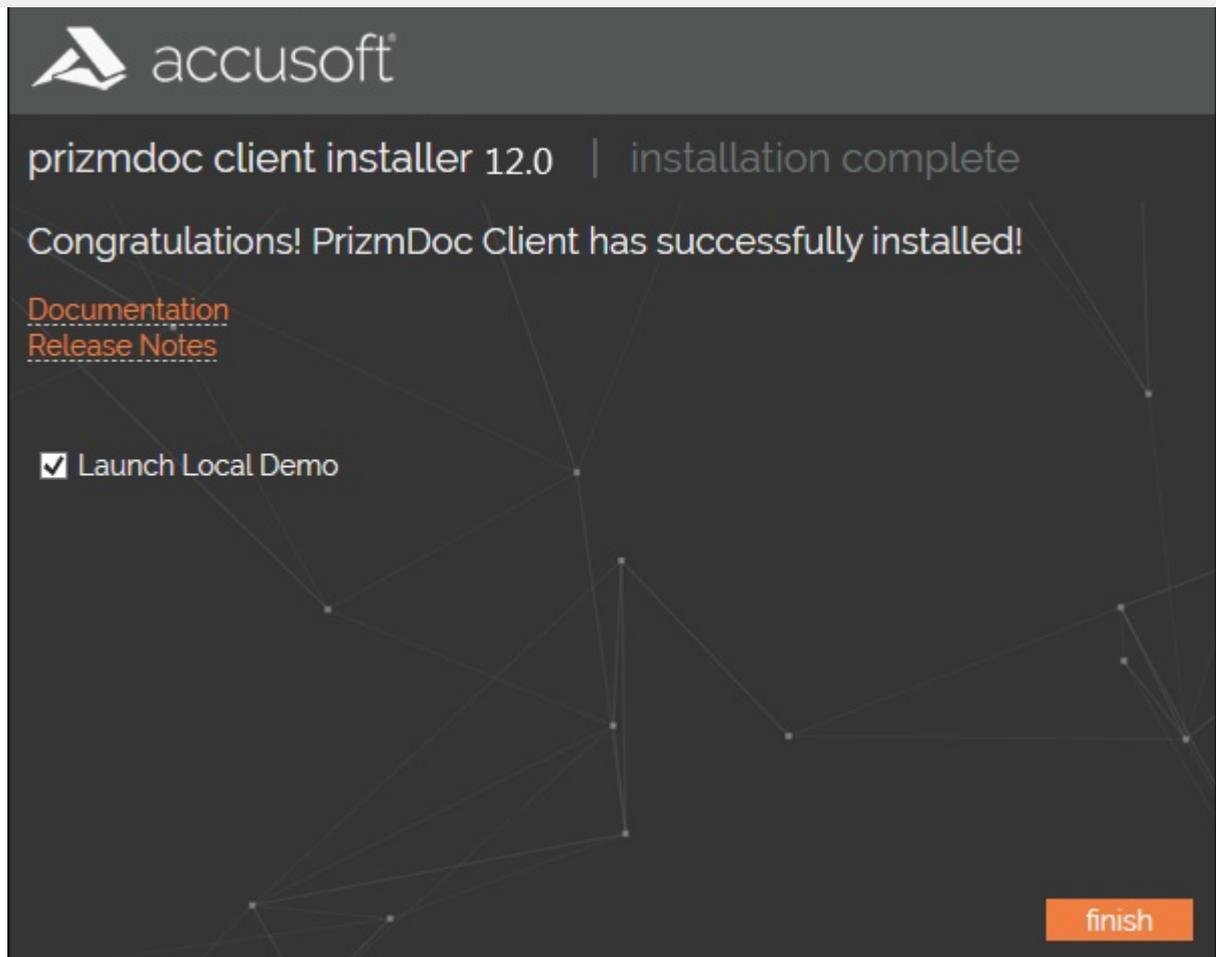
The screenshot shows a dark-themed dialog box for the PrizmDoc client installer 12.0. At the top left is the Accusoft logo. The title bar reads "prizmdoc client installer 12.0 | account information". Below the title bar, the text "Specify a login account for PrizmDoc Client Service:" is displayed. There are two input fields: "Account Name" and "Password". To the right of the "Account Name" field is a "browse..." button. At the bottom of the dialog, there are three buttons: "cancel", "back", and "install". The "install" button is highlighted in orange.

8. Once you have entered in all of the appropriate information, click **Install** to continue. (If the password is not correct an error dialog will be displayed noting that the password is not correct.) The Installation dialog is displayed with a progress bar:



The Installer will unpack the product, and lay out and configure the product as defined in the configuration dialogs. While PrizmDoc is installing, you can look at the **Release Notes**, **Online Demos**, or **Code Examples** from the links provided.

9. Once the installation is complete, the Installation Complete dialog is displayed:



There are links to the **Documentation** and product **Release Notes** if you wish to view them. There will also be an option to launch the locally installed demo in your default browser.

The option to launch the local demo is only available if all of the following occurred:

- Sample Documents and Code were installed.
- The Samples were configured and re-registered with IIS.
- The PrizmDoc Application Services were installed.

10. Click **Finish** to exit the Installer.

11. Go to [Check the Connection to Your Self-Hosted Server](#) for instructions on verifying that your installation of the PrizmDoc Viewer is able to contact the back-end services.

## Unattended Install & Uninstall

### Unattended Install

The PrizmDoc Viewer Windows installer can be installed unattended, however certain properties must be set:

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.	None
ServicePassword	Required - The password for the ServiceUser.	None
InstallFolder	Optional - The base installation directory for the product.	"C:\Prizm"
SelectedClientFeatures	Optional - The Viewer features to install. This can be empty or a comma separated list of values that can contain any of the following: <ul style="list-style-type: none"> <li>"HTML5Viewer"</li> <li>"LocalFileViewerFeature"</li> </ul>	"HTML5Viewer, LocalFileViewerFeature"
IISConfigureSamples	Optional - Whether to configure the samples with IIS or not. Set to "1" for yes and set to an empty string for no.	"1"
IISReregister	Optional - Whether to re-register ASP.NET v4 with IIS or not. Set to "1" for yes and set to an empty string for no.	"1"
SelectedPASFeatures	Optional - This can be set to "ALL" to include the PrizmDoc Application Services (PAS) features or set to an empty string to not include them.	"ALL"
PrizmScheme	Optional - The scheme that PrizmDoc Application Services will use for the self-hosted PrizmDoc server: <ul style="list-style-type: none"> <li>"HTTP"</li> <li>"HTTPS"</li> </ul> <p>Note: This value is required if PrizmHost or PrizmPort are defined.</p>	None
PrizmHost	Optional - The hostname or IP address for the self-hosted PrizmDoc server. <p>Note: This value is required if PrizmScheme or PrizmPort are defined.</p>	None
PrizmPort	Optional - The port that PrizmDoc Application Services will use for the self-hosted PrizmDoc server. <p>Note: This value is required if PrizmScheme or PrizmHost are defined.</p>	None

To start the unattended install:

1. The PrizmDocClient.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

and ServicePassword).

```
> PrizmDocClient.exe ServiceUser=accusoft.com\PrizmUser  
ServicePassword=mdppassword -s -l output.log
```

 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 4 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.

```
> start /wait PrizmDocClient.exe ServiceUser=accusoft.com\PrizmUser  
ServicePassword=mdppassword -s -l output.log
```

3. PrizmDoc should now be installed and started.

## Unattended Uninstall

You can use the -u flag as shown below to silently uninstall PrizmDoc on Windows:

```
> PrizmDocClient.exe -s -u -l output.log
```

## Install on Linux

This section contains the following information:

- [Linux Installation](#)
- [Configure Server Connections](#)
- [Uninstall PrizmDoc on Linux](#)

## Linux Installation

To install the PrizmDoc Viewer using a self-hosted PrizmDoc Server, follow the steps provided in this section:

Some steps are specific to a particular Linux distribution; these steps will be labeled as being specific to one of the following:

- Red Hat / Fedora (CentOS) Linux Distributions
- Debian (Ubuntu) Linux Distributions

## Step 1 - Download the PrizmDoc Viewer Linux Distribution

 Packages are only available for 64-bit systems.

Before downloading PrizmDoc, you will need to purchase a license key or request a Trial Evaluation by filling out the appropriate form at [www.accusoft.com](http://www.accusoft.com).

Once you have filled out the form for a Trial Evaluation, you can download PrizmDoc by doing one of the following:

- Following the link provided in the response email and selecting the desired Linux distribution.
- Downloading directly to the Linux server using the **wget** command for the specific distribution as shown below:

 **You must substitute the version of the package you are using in the code examples below.** For example, if you are using v11.0, then specify "11.0" instead of "<version>". If the version is a hot fix, you will need to also specify the hot fix number, for example, "11.0.1".

### Red Hat Enterprise Linux, CentOS (prior to v7) and Similar Older Linux Distributions

#### Example

```
wget
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_client_<version>.x86_64.rpm.tar.gz
```

### Red Hat Enterprise Linux, CentOS (v7 and later)

#### Example

```
wget
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_client_<version>.x86_64.rpm.tar.gz
```

### Debian and Ubuntu Linux Distributions

#### Example

```
wget
http://products.accusoft.com/PrizmDoc/<version>/prizmdoc_client_<version>.amd64.deb.tar.gz
```

### Generic .tar.gz Distribution

#### Example

```
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, Fedora, CentOS, and Older Linux Distributions

The following example is for Red Hat, Fedora, CentOS, and older Linux distributions:

#### Example

```
yum install --nogpgcheck *.rpm
```

## Debian (Ubuntu) Linux Distributions

The following example is for Debian (Ubuntu) Linux distributions:

### 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 command
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_client_*.tar.gz
cd prizmdoc_client_*
ls prizm-*.tar.gz | xargs -n1 tar xzf
cp -R prizm /usr/share/
```

3. Go to [Configure Server Connections](#) for instructions on how to configure the connection between:
  - Your Viewer web tier and PAS and,
  - between PAS and PrizmDoc Server.

## Configure Server Connections

### Configure the Connection Between Your Viewer Web Tier and PrizmDoc Application Services

Each sample contains a configuration file where you can configure the connection between your Viewer web tier and PrizmDoc Application Services. See the Configuration section at the bottom of the following pages for more information on configuring **PrizmApplicationServicesScheme**, **PrizmApplicationServicesHost**, and **PrizmApplicationServicesPort**.

- Getting Started with PrizmDoc > Integrating the Viewer with Your Application > Configuring the Viewer Samples > [PHP Sample](#)
- Getting Started with PrizmDoc > Integrating the Viewer with Your Application > Configuring the Viewer Samples > [JSP Sample](#)

### Configure the Connection Between PrizmDoc Application Services and PrizmDoc Server

Follow the steps below to configure the connection between PrizmDoc Application Services and PrizmDoc

1. On the machine where you installed PrizmDoc Application Services, locate the PrizmDoc Application Services configuration file. Assuming the standard install location, this is **`/usr/share/prizm/pas/pcc.nix.yml`**.
2. Edit the file and specify values for **`pccServer.hostName`**, **`pccServer.port`**, and **`pccServer.scheme`**.
  - **`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 single-server mode and 18682 by default for multi-server mode.
3. Restart PrizmDoc Application Services for the changes to take effect. See [Starting & Stopping Application Services](#) for instructions.
4. See the [Check the Connection to your Self-Hosted Server](#) page for instructions on verifying that your installation of PrizmDoc Application Services is able to contact the back-end services.

## Uninstall PrizmDoc on Linux

To uninstall PrizmDoc Viewer from your Linux system, perform the following steps:

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:

**Debian/Ubuntu:**

```
# sudo apt-get remove prizm-contentconnect prizm-pas
```

**Red Hat/CentOS:**

```
# yum remove prizm-contentconnect prizm-pas
```

**Generic Package:**

This will also remove PrizmDoc Server if it is installed.

```
# rm -Rf /usr/share/prizm
```

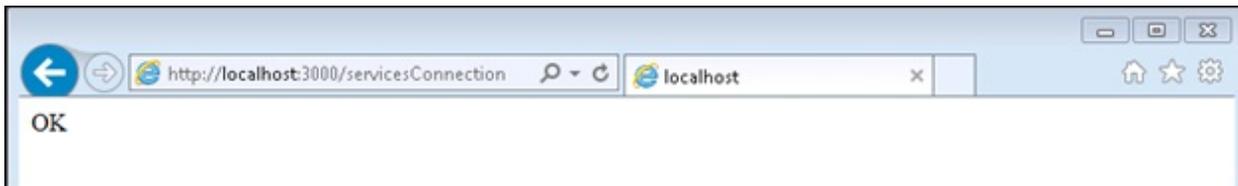
3. There will potentially be temporary files left behind. The same command to remove the generic package can be used to remove all remaining files. This will also remove PrizmDoc Server if it is installed.

```
# rm -Rf /usr/share/prizm
```

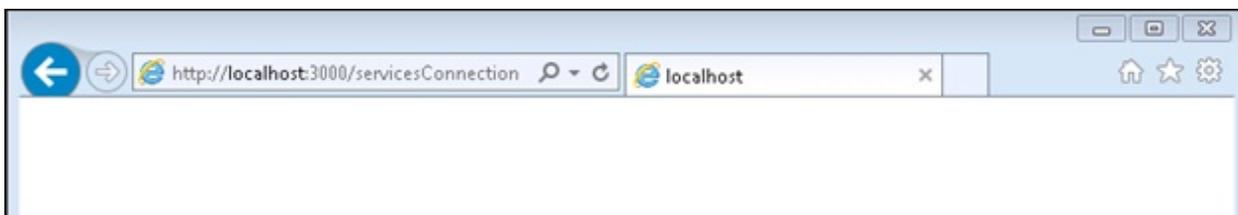
## Check the Connection to your Self-Hosted Server

Application Services (PAS) to check your connection to PrizmDoc Server.

1. On the machine where you installed PrizmDoc Application Services, 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 PrizmDoc Application Services.
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:



For more information, refer to the [Troubleshooting](#) section.

3. Once you are done checking the connection, go to [Step 3 - Integrate the Viewer with Your Application](#).

## Set up Viewer Samples (Optional)

If you want to view samples that we have provided, you can follow the steps below. The content in this topic is optional and not required.

The PrizmDoc Server samples demonstrate document viewing via the Viewer communicating via a web server with PrizmDoc Server using a RESTful scheme as noted in the [PrizmDoc Server RESTful API](#) topic. The samples show the web service layer being implemented in either Microsoft C#, PHP, or Java.

The following steps summarize the events required to initiate a document viewing session between PrizmDoc Server and the web server:

1. Web server requests a new viewing session from PrizmDoc Server:

### Example

```
POST http://localhost:18681/PCCIS/V1/ViewingSession
```

PrizmDoc Server responds with the Document/Session Identification that the web server is to use for all further PrizmDoc Server service communication for the loaded document.

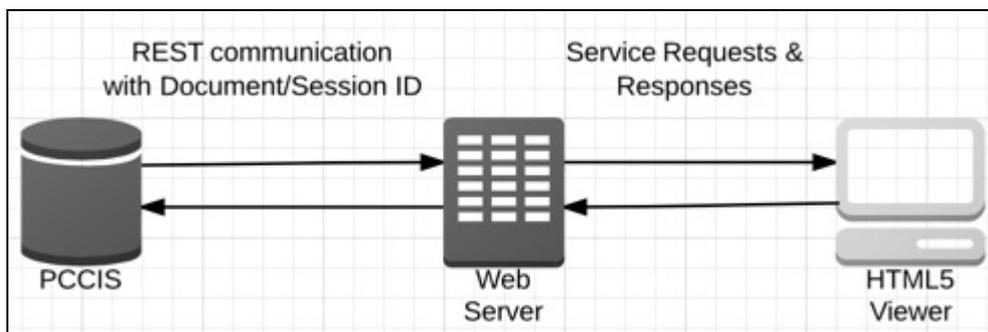
## Example

```
PUT http://localhost:18681/PCCIS/V1/ViewingSessions/u{Viewing Session ID}/SourceFile?FileExtension={File Extension}
```

3. Web server initiates a viewing Session in PrizmDoc Server:

## Example

```
POST http://localhost:18681/PCCIS/V1/ViewingSessions/u{Viewing Session ID}/Notification/SessionStarted
```



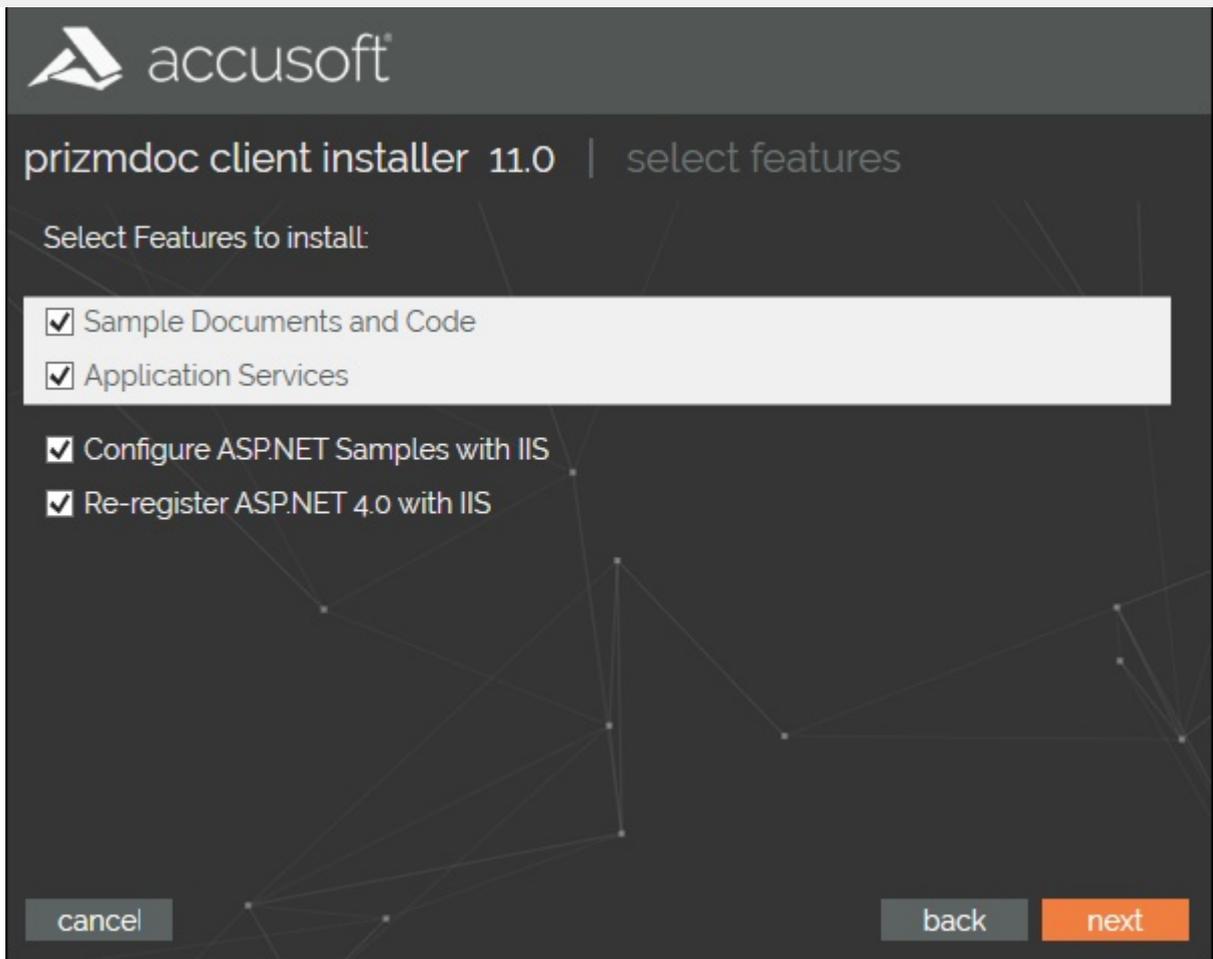
Samples:

- [.NET WebForms Sample](#)
- [.NET MVC 5 Sample](#)
- [PHP Sample](#)
- [JSP Sample](#)

## .NET WebForms Sample

### Installation

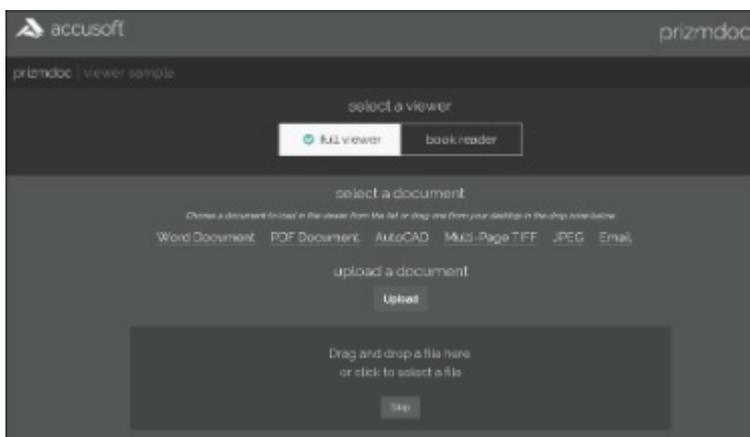
1. Prior to installation, ensure Microsoft's Internet Information Service (IIS) and ASP.NET 4.0+ are enabled on the computer that will be running the .NET Web Forms sample.
2. Begin the installation of [PrizmDoc for Windows](#).
3. During the installation of PrizmDoc, make sure to select the following options:
  - **Sample Documents and Code** – Installs the different client viewers that can be used with the PrizmDoc services and sample documents.
  - **Application Services** - Installs the service that provides application-level logic for the Viewer, such as enabling document viewing through the PrizmDoc Server, saving and loading of markup, and handling opening of documents and creating viewing sessions.
  - **Configure ASP.NET Samples with IIS.**
  - **Re-register ASP.net 4.0 with IIS.**



4. After the installation, test the sample application in a browser:

The following will route you directly to the Viewer sample splash page:

[http://localhost:18000/PrizmDoc\\_HTML5\\_Viewer\\_NET\\_WEBFORMS/](http://localhost:18000/PrizmDoc_HTML5_Viewer_NET_WEBFORMS/)



5. From the splash page you have two options:

Choice of Viewer:

Select a sample document -OR- upload a document:

- You can choose any of the 5 sample documents (Word, PDF, CAD, Tiff, or JPEG).
- 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.

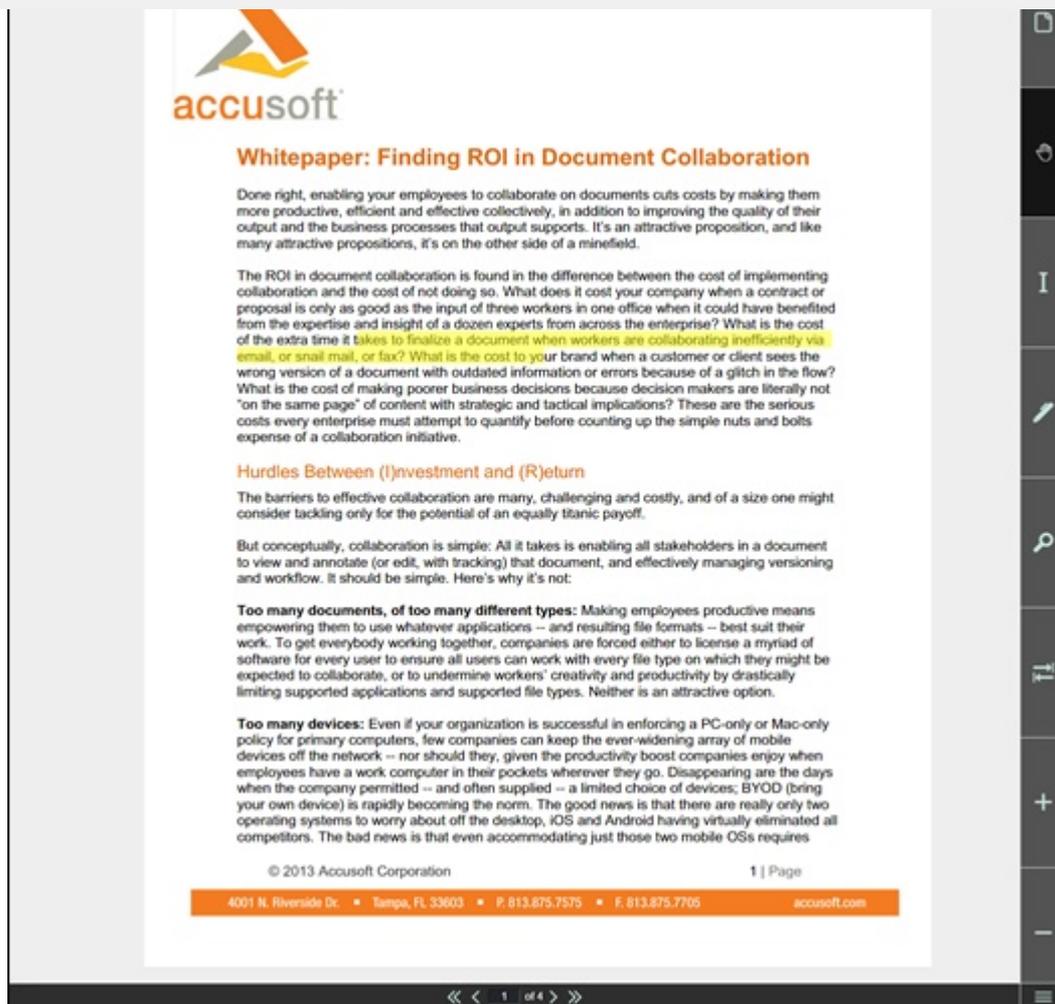
## 6. Full Viewer

If you select **Full Viewer** on the splash page, then documents will be viewed with the full-featured, out-of-the-box responsive Viewer:



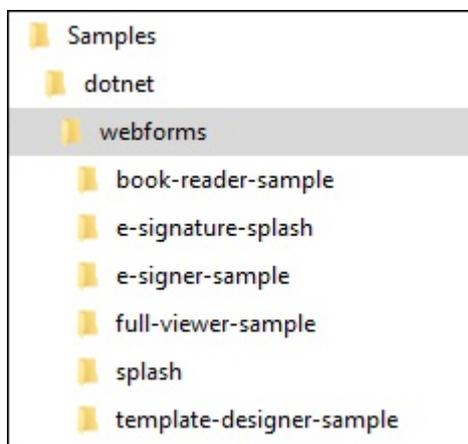
## 7. Book Reader

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:



## Directory Structure

The samples are installed under 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

## Folder contents: full-viewer-sample

File / Folder	Description
App_Code folder	<p>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. See PrizmApplicationServices.cs to see how we integrate the sample with PrizmDoc Application Services and see PccConfig.cs to see how we load the pcc.config configuration file.</p> <p>The files in this folder are essential and must be re-distributed to run the full Viewer.</p>
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the Viewer.
viewer-assets/less 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.
viewer-assets/Gulpfile.js	Contains Gulp tasks to build the viewer less and icons. 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. 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 the PrizmDoc Application Services.
viewer-webtier/pcc.ashx	This file handles all incoming requests from the Viewer. This file simply uses the App_Code/PrizmApplicationServices.cs class to forward all requests to PrizmDoc Application Services
viewer-webtier/pcc.config	Defines the connection settings for the PrizmDoc Application Services.
Default.aspx, Default.aspx.cs	The default page for the sample. This page loads the full Viewer.
web.config	Contains IIS settings.
predefinedSearch.json	<p>This data file contains information defining search queries that will appear as selectable items in the full Viewer.</p> <p>Note: This file is consumed by the page Default.aspx and the JSON is injected into the HTML that is returned by Default.aspx. Ultimately, the predefined search terms are provided as a JavaScript hash, when the Viewer is created.</p>
redactionReason.json	<p>This data file contains information defining redaction reasons that are available in the Viewer.</p> <p>Note: This file is consumed by the page Default.aspx and the JSON is injected into the HTML that is returned by Default.aspx. Ultimately, the redaction reasons are provided as a JavaScript hash, when the Viewer is created.</p>
Global.asax	The Global.asax 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

ASP.NET-based application. We use this file to initialize our PccConfig 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 PrizmDoc Application Services. While the code for the classes can be modified as needed, modifications should be done with care. See PrizmApplicationServices.cs to see how we integrate the sample with PrizmDoc Application Services and see PccConfig.cs to see how we load the pcc.config 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.
viewer-assets/Gruntfile	Contains Grunt tasks to build the reader less. 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 ( <a href="https://icomoon.io">https://icomoon.io</a> ) 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 the PrizmDoc Application Services.
viewer-webtier/pcc.ashx	This file handles all incoming requests from the Viewer. This file simply uses the App_Code/PrizmApplicationServices.cs class to forward all requests to PrizmDoc Application Services
viewer-webtier/pcc.config	Defines the connection settings for the PrizmDoc Application Services.
index.html	The default page for the sample. This page calls the pcc.ashx handler to start a viewing session with the PrizmDoc Application Services 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 Global.asax file, also known as the ASP.NET application file, is a file that

	HttpModules. The Global.asax file resides in the root directory of an ASP.NET-based application. We use this file to initialize our PccConfig 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 PrizmDoc Application Services. While the code for the classes can be modified as needed, modifications should be done with care. See PrizmApplicationServices.cs to see how we integrate the sample with PrizmDoc Application Services and see PccConfig.cs to see how we load the pcc.config 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 viewer-assets/js/bundle.js and viewer-assets/css/bundle.css by the build process defined in Gulpfile.js.  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-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 assets. 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 the PrizmDoc Application Services.
viewer-webtier/pcc.ashx	This file handles all incoming requests from the Viewer. This file simply uses the App_Code/PrizmApplicationServices.cs class to forward all requests to PrizmDoc Application Services
viewer-webtier/pcc.config	Defines the connection settings for the PrizmDoc Application Services.
index.html	The default page for the sample. This page calls the pcc.ashx handler to start a viewing session with the PrizmDoc Application Services 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 bundle.js and bundle.css that are found in the viewer-assets folder.
Global.asax	The Global.asax 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

	based application. We use this file to initialize our PccConfig class.
e-signer-sample.sln	Visual Studio solution file to open the sample.

## Folder contents: template-designer-sample

File / Folder	Description
App_Code folder	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. See PrizmApplicationServices.cs to see how we integrate the sample with PrizmDoc Application Services and see PccConfig.cs to see how we load the pcc.config 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 viewer-assets/js/bundle.js and viewer-assets/css/bundle.css by the build process defined in Gulpfile.js.  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-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 assets. 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 the PrizmDoc Application Services.
viewer-webtier/pcc.ashx	This file handles all incoming requests from the Viewer. This file simply uses the App_Code/PrizmApplicationServices.cs class to forward all requests to PrizmDoc Application Services
viewer-webtier/pcc.config	Defines the connection settings for the PrizmDoc Application Services.
index.html	The default page for the sample. This page calls the pcc.ashx handler to start a viewing session with the PrizmDoc Application Services 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 bundle.js and bundle.css that are found in the viewer-assets folder.
Global.asax	The Global.asax 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

based application. We use this file to initialize our PccConfig class.

template-designer- Visual Studio solution file to open the sample.  
sample.sln

## Configuration with pcc.config

The file **pcc.config** is used to configure the connection settings between the web tier and PrizmDoc Application Services. 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.

<DocumentPath> (Only in splash pages)	The sample pulls named documents from this location. The DocumentPath must have read/write permissions in order for the file drag and drop functionality of the splash page to work.
<PrizmApplicationServices[Scheme Host Port]>	Specifies how to connect to the PrizmDoc Application Services

## Development Information

The C# sample has the following requirements for development:

- Visual Studio v2010 and later
- .NET 4.0 and later

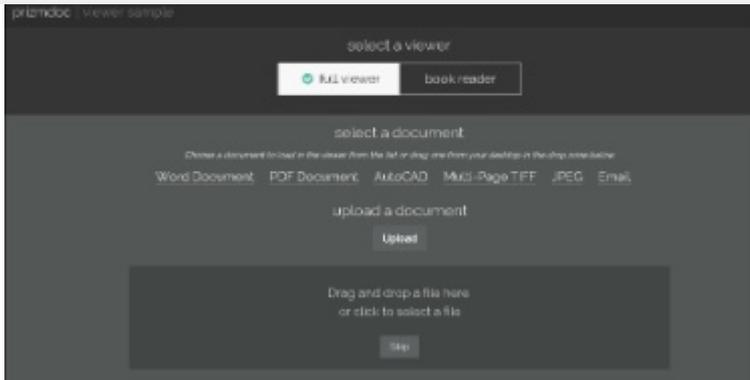
## .NET MVC 5 Sample

### Installation

1. Ensure that Microsoft's Internet Information Service (IIS) is enabled ([http://msdn.microsoft.com/en-us/library/ms181052\(v=vs.80\).aspx](http://msdn.microsoft.com/en-us/library/ms181052(v=vs.80).aspx)) on the computer that will be running the .NET MVC 5 sample.
2. Ensure that .NET 4.5 is installed. You can download it from this page: <https://www.microsoft.com/en-us/download/details.aspx?id=30653>.
3. Begin the installation of **PrizmDoc for Windows**.
4. After the installation, test the sample application in a browser:

The following will route you directly to the Viewer sample splash page:

[http://localhost:18000/PrizmDoc\\_HTML5\\_Viewer\\_NET\\_MVC](http://localhost:18000/PrizmDoc_HTML5_Viewer_NET_MVC)



5. From the splash page you have two options:

Choice of viewer:

- You can choose to load either the **Full Viewer** or the **Book Reader**

Select a sample document -OR- upload a document:

- You can choose any of the 5 sample documents (Word, PDF, CAD, Tiff, or JPEG)
- 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.

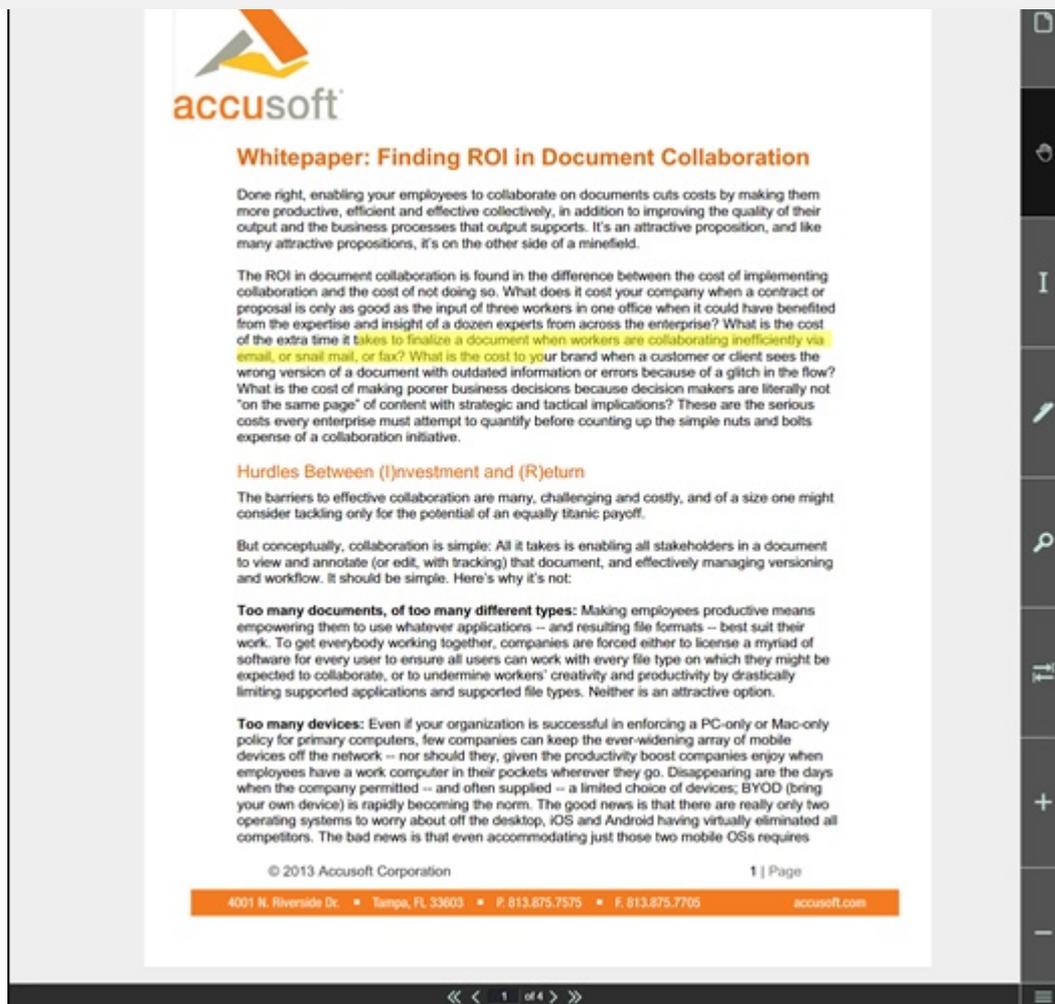
6. Full Viewer:

If you select **Full Viewer** on the splash page, then documents will be viewed with the full-featured, out-of-the-box Viewer:



## 7. 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:



## .NET MVC 5 Sample Directory Structure

The .NET MVC 5 sample is installed under 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.

 If you are interested in seeing how we handle the requests, please take a look at the source code in Modes/PccViewer.

File / Folder	Description
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the Viewer.
viewer-assets/less 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.
viewer-assets/Gulpfile.js	Contains Gulp tasks to build the viewer less and icons. This file is non-essential and does not need to be re-distributed.
viewer-assets/package.json	A file used by <b>npm</b> (a package manager). It defines the dependencies installed by npm, which are required to run Grunt and compile the less.
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 Default.aspx and the JSON is injected into the HTML that is returned by Default.aspx. 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 Default.aspx and the JSON is injected into the HTML that is returned by Default.aspx. Ultimately, the redaction reasons are provided as a JavaScript hash, when the Viewer is created.

## Folder contents: viewers/book-reader-sample

File / Folder	Description
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the book reader viewer.
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.
viewer-assets/Gruntfile	Contains Grunt tasks to build the reader less. This file is non-essential and does not need to be re-distributed.
viewer-assets/package.json	A file used by <b>npm</b> (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 ( <a href="https://icomoon.io">https://icomoon.io</a> ) to generate a new icon font. This file is non-essential and does not need to be re-distributed.
viewer-assets/js/sample-config.js	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
---------------	-------------

	<p>assets/js/bundle.js and viewer-assets/css/bundle.css by the build process defined in Gulpfile.js.</p> <p>The files in this folder are non-essential and do not need to be re-distributed.</p>
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-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 assets. This file is non-essential and does not need to be re-distributed.
index.html	The default page for the sample. This page calls the pcc.ashx handler to start a viewing session with the PrizmDoc Application Services and then the page loads the Viewer.
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 bundle.js and bundle.css that are found in the viewer-assets folder.

## Folder contents: viewers/template-designer-sample

File / Folder	Description
modules folder	<p>Contains uncompiled assets of the Viewer. These files will be compiled to viewer-assets/js/bundle.js and viewer-assets/css/bundle.css by the build process defined in Gulpfile.js.</p> <p>The files in this folder are non-essential and do not need to be re-distributed.</p>
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-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 assets. This file is non-essential and does not need to be re-distributed.
index.html	The default page for the sample. This page calls the pcc.ashx handler to start a viewing session with the PrizmDoc Application Services and then the page loads the Viewer.
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 bundle.js and bundle.css that are found in the viewer-assets 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

<DocumentPath>	The sample pulls named documents from this location. The DocumentPath must have read/write permissions in order for the file drag and drop functionality of the splash page to work.
<PrizmApplicationServices[Scheme Host Port]>	Specifies how to connect with the PrizmDoc Application Services.

## Development Information

The MVC 5 sample has the following requirements for development:

- Visual Studio v2012 and later
- .NET 4.5 and later

## PHP Sample

### Supported PHP Versions

The PrizmDoc PHP Sample requires PHP version 5.4 and above.

### Linux Installation

1. Install [PrizmDoc for Linux](#) and verify the service is working correctly. The installation will place the PHP samples in the following directory: **/usr/share/prizm/Samples/php**.
2. Install Apache and PHP. Refer to the directions that are specific to your distribution.
3. Add the following to the Apache configuration:

#### Example

```
Alias /pccis_sample /usr/share/prizm/Samples/php
<Directory /usr/share/prizm/Samples/php>
AllowOverride All
Require all granted
</Directory>
```

4. On the command line, first check that the configuration file validates:

#### Example

```
apache2ctl configtest
```

5. Then restart the Apache service to read the configuration changes:

#### Example

6. Give read/write permissions to the Documents folder and the Markups folder. Give read permissions to the ImageStamp folder. These folders are installed to: /usr/share/prizm/Samples on Linux. For more information, see the "Configuration with pcc.config" section below.
7. Test the sample application in a browser. The following will route you directly to the Viewer sample splash page: `http://myservername:port/pccis_sample/splash`

## Install Considerations

### Ubuntu

For Ubuntu users: In Ubuntu 13 and up, the `php_curl` module was unbundled from the default PHP install. This module is required by the PHP sample, and should be installed separately, as such:

#### Example

```
sudo apt-get install php5-curl
```

### Apache

Depending on how you install PHP for Apache, you may need to install the Apache PHP addition, as such:

#### Example

```
sudo apt-get install libapache2-mod-php5
```

### Configure PHP Web Tier with SELinux

When running PHP Web Tier on Security-Enhanced Linux you may experience issues with loading documents in the Viewer, and may see the following errors in Apache's log files: **PHP Warning: file\_get\_contents(<URL>): failed to open stream: Permission denied**

The reason for the error is because the Web Tier needs to contact the PrizmDoc Server. By default SELinux disallows Apache processes from connecting out. You can run the following command to allow the web tier to function:

#### Example

```
setsebool -P httpd_can_network_connect 1
```

Similarly, when attempting to upload a document, the PHP application may appear unresponsive. When reviewing the document upload response in your browser's console, you may see a 403 Forbidden error. Security-Enhanced Linux prevents the PHP / Apache HTTP application from writing to the file system by default. You can run the following command to set the **httpd\_sys\_rw\_content\_t** policy for the Samples directory:

#### Example

```
chcon -R -t httpd_sys_rw_content_t /usr/share/prizm/Samples/
```

## Windows Installation

By default, the PHP sample is installed to: **C:\Prizm\Samples\php**, assuming use of the default PrizmDoc

## IIS

Install IIS, along with the CGI component, and then install and configure PHP. The instructions and installer provided on the IIS site are now old and outdated, as they install PHP version 5.3. Instead, install the latest version of PHP using the Web Platform Installer. If using an already configured server, make sure you are using PHP version 5.4 or higher.

To configure the sample:

1. Note the **location** of the PHP sample as described above. If you changed the PrizmDoc default install location, note the comparable sample in the install location you specified.
2. Open the **IIS Manager**.
3. In the Sites list, select the **site** you want to add the PHP sample to, such as "Default Web Site". Right-click on it and select "**Add Virtual Directory...**".
4. In the Alias field, type the **URL path** you want to use for the sample. We will use "**pccis\_sample**" as an example here.
5. In the Physical Path field, use the location we noted above. Click **OK** to create the virtual directory.

Now you can browse to [http://localhost/pccis\\_sample/splash](http://localhost/pccis_sample/splash) in your browser to see the PHP sample.

## Apache

Install Apache, or an Apache stack like XAMPP or WAMP server, and a supported version of PHP.

To configure the sample:

1. Note the **location** of the PHP sample as described above. If you changed the PrizmDoc default install location, note the comparable sample in the install location you specified.
2. Open the **httpd.conf** file of your Apache install.
3. In the block starting with "**<IfModule alias\_module>**", add the following to the end of the block:

### Example

```
# PCC PHP sample alias
Alias /pccis_sample "C:/Prizm/Samples/php"
<Directory "C:/Prizm/Samples/php">
  Allow from all
  Require all granted
</Directory>
```

4. Restart **Apache** so that it picks up the new settings.

Now you can browse to [http://localhost/pccis\\_sample/splash](http://localhost/pccis_sample/splash) in your browser to see the PHP sample.

## Optional Configuration

By default, PHP has a small file size upload. If you need to be able to upload larger files through the splash page, you may need to change the following values in the php.ini file of your PHP installation:

```
upload_max_filesize = 1000M
post_max_size = 1000M
```

This sets the file upload to 1GB, which should be enough to test most files through the splash page. These settings may be too high for a production service, so adjust those accordingly when deploying to production.

## Configure PHP Web Tier with SELinux

When running PHP Web Tier on Security-Enhanced Linux you may experience issues with loading documents in the Viewer, and may see the following errors in Apache's log files: **PHP Warning: file\_get\_contents(<URL>): failed to open stream: Permission denied**

The reason for this is that the Web Tier needs to contact the PrizmDoc Server. By default SELinux disallows Apache processes from connecting out. You can run the following command to allow the web tier to function:

### Example

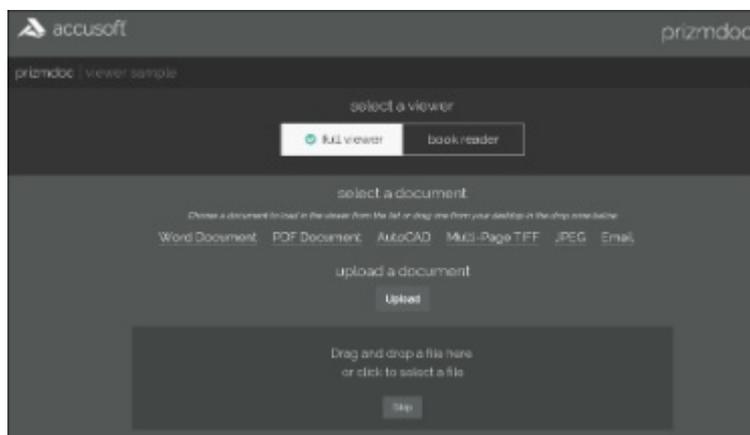
```
setsebool -P httpd_can_network_connect 1
```

## Browsing the Sample

From the splash page you have two options:

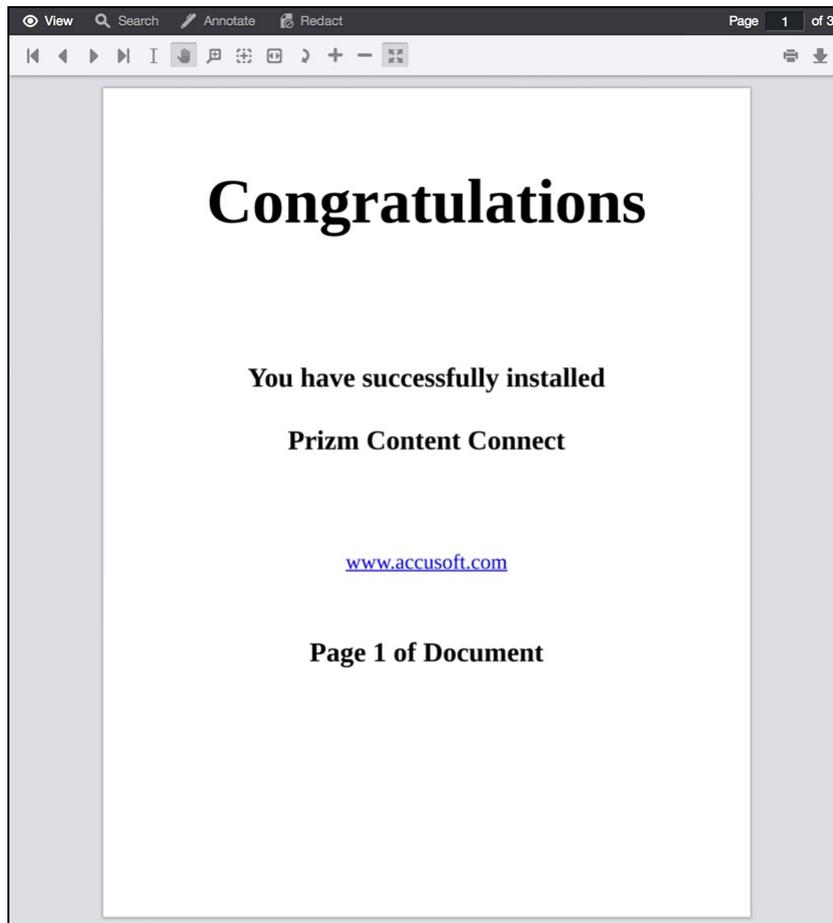
1. Choice of Viewer: you can choose to load either the **Full Viewer** or the **Book Reader**.
2. Select a sample document - OR - upload a document:

You can choose any of the 5 sample documents (Word, PDF, CAD, Tiff or JPEG). 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.



3. Full Viewer:

out-of-the-box responsive viewer.



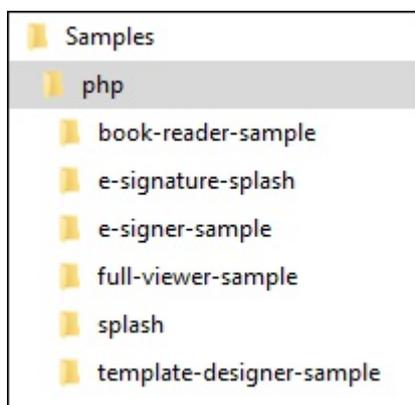
#### 4. Book Reader:

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:



## Directory Structure

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

sample folders contain only the files needed to run the sample.

## Folder contents: full-viewer-sample

File / Folder	Description
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the Viewer.
viewer-assets/less 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.
viewer-assets/Gulpfile.js	Contains Gulp tasks to build the viewer less and icons. This file is non-essential and does not need to be re-distributed.
viewer-assets/package.json	A file used by <b>npm</b> (a package manager). It defines the dependencies installed by npm, which are required to run Grunt and compile the less.
viewer-webtier folder	Contains files that implement the php layer of communication between the Viewer and the PrizmDoc Application Services.
viewer-webtier/pas.php	Handles all requests from the Viewer and forwards them to the PrizmDoc Application Services.
viewer-webtier/pcc.config	Defines the connection settings for the PrizmDoc Application Services.
viewer-webtier/PccViewer/Config.php	This class parses the <b>pcc.config</b> file and provides methods for easily reading its parameters.
index.php	The default web starting page in this sample. The Viewer's code gets loaded by this page.
.htaccess	Contains Apache web server 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 Default.aspx and the JSON is injected into the HTML that is returned by Default.aspx. 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 Default.aspx and the JSON is injected into the HTML that is returned by Default.aspx. Ultimately, the redaction reasons are provided as a JavaScript hash, when the Viewer is created.

## Folder contents: book-reader-sample

File / Folder	Description
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the book reader Viewer.

	non-essential, and does not need to be re-distributed.
viewer-assets/Gruntfile.js	Contains Grunt tasks to build the viewer less. This file is non-essential and does not need to be re-distributed.
viewer-assets/package.json	A file used by <b>npm</b> (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 ( <a href="https://icomoon.io">https://icomoon.io</a> ) 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 php layer of communication between the book reader Viewer and the PrizmDoc Application Services.
viewer-webtier/pas.php	Handles all requests from the Viewer and forwards them to the PrizmDoc Application Services.
viewer-webtier/pcc.config	Defines the connection settings for the PrizmDoc Application Services.
viewer-webtier/PccViewer/Config.php	This class parses the <b>pcc.config</b> file and provides methods for easily reading its parameters.
index.html	The default page for the sample. This page loads the Viewer.
sample-config.js	Contains references to the assets, web tier, and language files used by the Viewer in this sample.
.htaccess	Contains Apache web server settings.

## Folder contents: e-signer-sample

File / Folder	Description
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the book reader Viewer.
modules folder	Contains uncompiled assets of the Viewer. These files will be compiled to viewer-assets/js/bundle.js and viewer-assets/css/bundle.css by the build process defined in Gulpfile.js.  The files in this folder are non-essential and do not need to be re-distributed.
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-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 assets. This file is non-essential and does not need to be re-distributed.
viewer-webtier folder	Contains files that implement the php layer of communication between the book reader Viewer and the PrizmDoc Application Services.
viewer-webtier/pas.php	Handles all requests from the Viewer and forwards them to the PrizmDoc Application Services.

viewer-webtier/PccViewer/Config.php	This class parses the <b>pcc.config</b> file and provides methods for easily reading its parameters.
index.html	The default page for the sample. This page loads the Viewer.
.htaccess	Contains Apache web server 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 bundle.js and bundle.css that are found in the viewer-assets folder.

## Folder contents: template-designer-sample

File / Folder	Description
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the book reader Viewer.
modules folder	Contains uncompiled assets of the Viewer. These files will be compiled to viewer-assets/js/bundle.js and viewer-assets/css/bundle.css by the build process defined in Gulpfile.js.  The files in this folder are non-essential and do not need to be re-distributed.
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-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 assets. This file is non-essential and does not need to be re-distributed.
viewer-webtier folder	Contains files that implement the php layer of communication between the book reader Viewer and the PrizmDoc Application Services.
viewer-webtier/pas.php	Handles all requests from the Viewer and forwards them to the PrizmDoc Application Services.
viewer-webtier/pcc.config	Defines the connection settings for the PrizmDoc Application Services.
viewer-webtier/PccViewer/Config.php	This class parses the <b>pcc.config</b> file and provides methods for easily reading its parameters.
index.html	The default page for the sample. This page loads the Viewer.
.htaccess	Contains Apache web server 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 bundle.js and bundle.css that are found in the viewer-assets folder.

## Configuration with pcc.config

The file **pcc.config** is used to configure the connection settings between the web tier and PrizmDoc

is self-documenting, but a little information about the configuration options is given below.

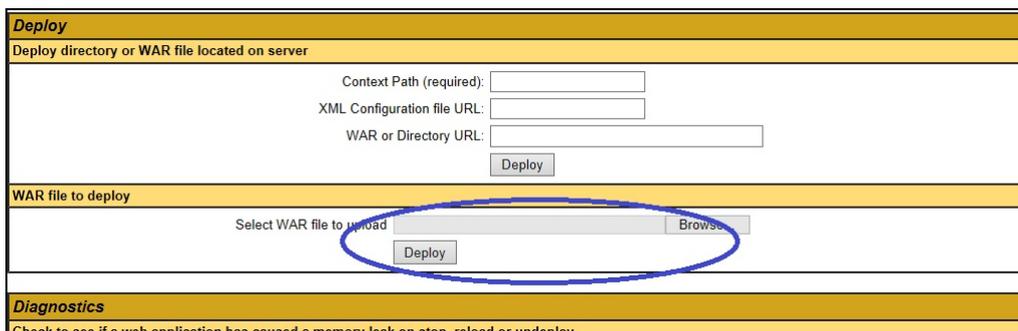
<DocumentPath> (Only in splash pages)	The sample pulls named documents from this location. The DocumentPath must have read/write permissions in order for the file drag and drop functionality of the splash page to work.
<PrizmApplicationServices[Scheme Host Port]>	Specifies how to connect to the PrizmDoc Application Services

## JSP Sample

This topic contains steps for how to install on Linux and Windows.

 Note that JDK 1.7 and JRE 1.7+ are required.

1. Begin the installation of [PrizmDoc for Windows](#) or [PrizmDoc for Linux](#) depending on your operating system.
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:
  - For Windows users: **C:\Prizm\Samples\jsp\target**
  - For Linux users: **/usr/share/prizm/Samples/jsp/target**
6. Click **Deploy**:



The screenshot shows the Tomcat Manager interface. The 'Deploy' section is highlighted in yellow. It contains three input fields: 'Context Path (required):', 'XML Configuration file URL:', and 'WAR or Directory URL:'. Below these is a 'Deploy' button. The 'WAR file to deploy' section is also highlighted in yellow and contains a text input field with the placeholder 'Select WAR file to upload' and a 'Browse...' button. A blue oval is drawn around the 'Select WAR file to upload' text and the 'Deploy' button below it.

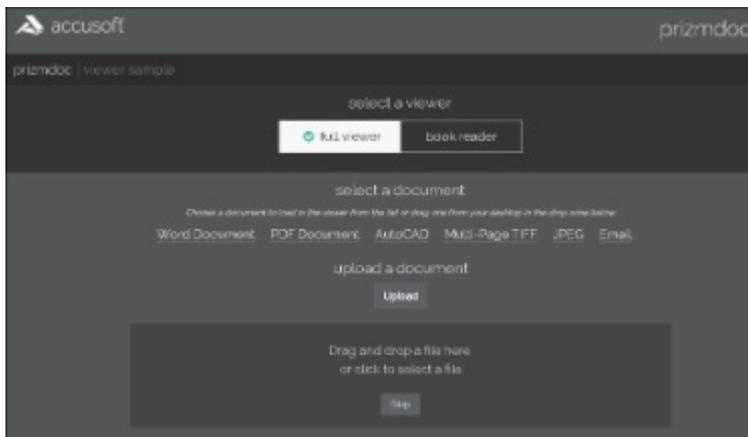
7. PrizmDoc Server web tier will be deployed on your Tomcat server. The deployed app may be found in the following location:
  - For Windows Users: **C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps**
  - For Linux Users: **/var/lib/tomcat7/webapps**
8. Give read/write permissions to the Documents folder and the Markups folder. Give read permissions to the ImageStamp folder. These folders are installed on the following location:

- For Linux Users: `/usr/share/prizm/Samples/jsp/target`

## Overview of Samples

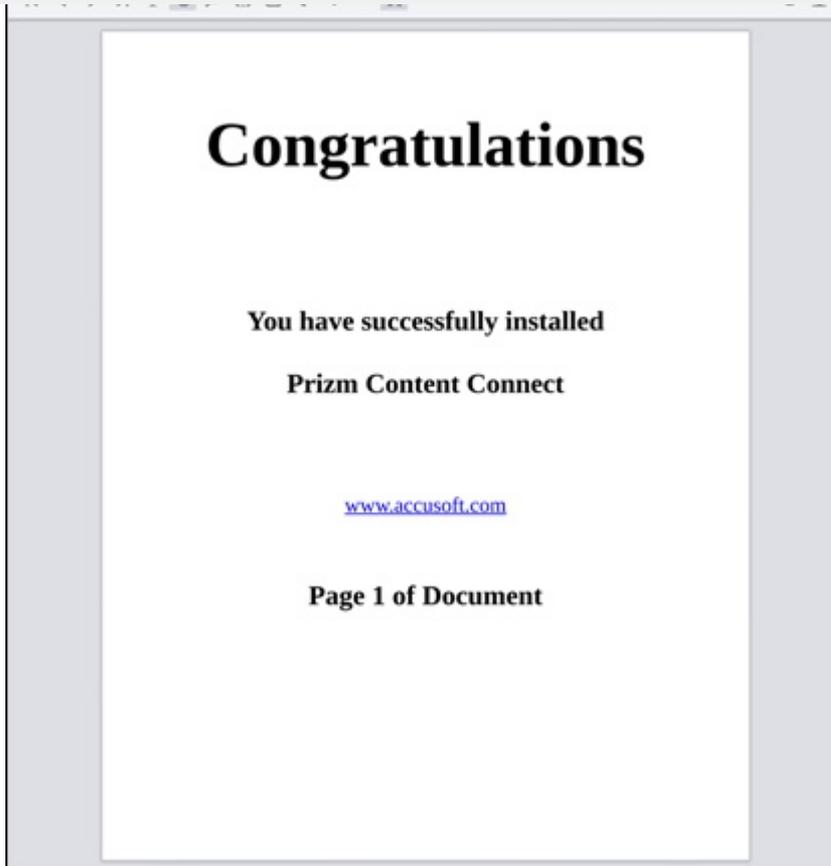
From the splash page you have two options:

1. Choice of Viewer:
  - You can choose to load either the **Full Viewer** or the **Book Reader**.
2. Select a sample document -OR- upload a document:
  - You can choose any of the 5 sample documents (Word, PDF, CAD, Tiff, or JPEG).
  - 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.



### 3. 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:



#### 4. 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:

**accusoft**

## Whitepaper: Finding ROI in Document Collaboration

Done right, enabling your employees to collaborate on documents cuts costs by making them more productive, efficient and effective collectively, in addition to improving the quality of their output and the business processes that output supports. It's an attractive proposition, and like many attractive propositions, it's on the other side of a minefield.

The ROI in document collaboration is found in the difference between the cost of implementing collaboration and the cost of not doing so. What does it cost your company when a contract or proposal is only as good as the input of three workers in one office when it could have benefited from the expertise and insight of a dozen experts from across the enterprise? What is the cost of the extra time it takes to finalize a document when workers are collaborating inefficiently via email, or snail mail, or fax? What is the cost to your brand when a customer or client sees the wrong version of a document with outdated information or errors because of a glitch in the flow? What is the cost of making poorer business decisions because decision makers are literally not "on the same page" of content with strategic and tactical implications? These are the serious costs every enterprise must attempt to quantify before counting up the simple nuts and bolts expense of a collaboration initiative.

### Hurdles Between (I)nvestment and (R)eturn

The barriers to effective collaboration are many, challenging and costly, and of a size one might consider tackling only for the potential of an equally titanic payoff.

But conceptually, collaboration is simple: All it takes is enabling all stakeholders in a document to view and annotate (or edit, with tracking) that document, and effectively managing versioning and workflow. It should be simple. Here's why it's not:

**Too many documents, of too many different types:** Making employees productive means empowering them to use whatever applications – and resulting file formats – best suit their work. To get everybody working together, companies are forced either to license a myriad of software for every user to ensure all users can work with every file type on which they might be expected to collaborate, or to undermine workers' creativity and productivity by drastically limiting supported applications and supported file types. Neither is an attractive option.

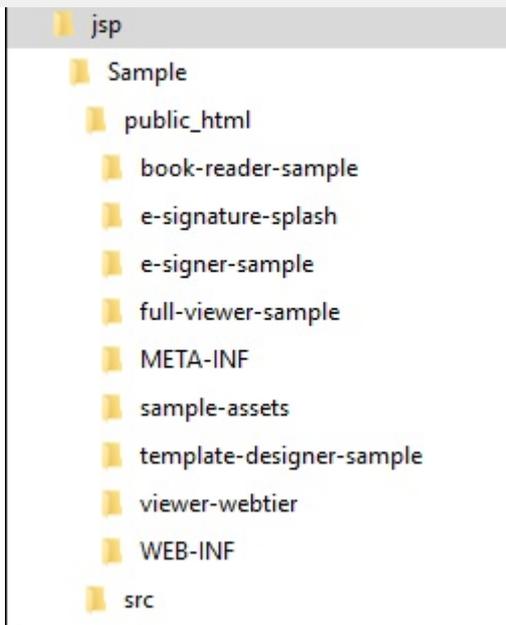
**Too many devices:** Even if your organization is successful in enforcing a PC-only or Mac-only policy for primary computers, few companies can keep the ever-widening array of mobile devices off the network – nor should they, given the productivity boost companies enjoy when employees have a work computer in their pockets wherever they go. Disappearing are the days when the company permitted – and often supplied – a limited choice of devices; BYOD (bring your own device) is rapidly becoming the norm. The good news is that there are really only two operating systems to worry about off the desktop, iOS and Android having virtually eliminated all competitors. The bad news is that even accommodating just those two mobile OSs requires

© 2013 Accusoft Corporation 1 | Page

4001 N. Riverside Dr. • Tampa, FL 33603 • P. 813.875.7575 • F. 813.875.7705 [accusoft.com](http://accusoft.com)

## 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 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 <b>Configuration with web.xml</b> section below.

## Folder contents: full-viewer-sample

File / Folder	Description
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the Viewer.
viewer-assets/less 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.
viewer-assets/Gulpfile.js	Contains Gulp tasks to build the viewer less and icons. This file is non-essential and does not need to be re-distributed.
viewer-assets/package.json	A file used by <b>npm</b> (a package manager). It defines the dependencies installed by npm, which are required to run Grunt and compile the less.
viewer-webtier folder	Contains files that implement the jsp layer of communication between the Viewer and the PrizmDoc Application Services.

webtier/pas.jsp	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 index.jsp and the JSON is injected into the HTML that is returned by index.jsp. 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 Default.aspx and the JSON is injected into the HTML that is returned by Default.aspx. Ultimately, the redaction reasons are provided as a JavaScript hash, when the Viewer is created.

## Folder contents: book-reader-sample

File / Folder	Description
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the book reader Viewer.
viewer-assets/less 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.
viewer-assets/Gruntfile.js	Contains Grunt tasks to build the viewer less. This file is non-essential and does not need to be re-distributed.
viewer-assets/package.json	A file used by <b>npm</b> (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 ( <a href="https://icomoon.io">https://icomoon.io</a> ) 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 jsp layer of communication between the book reader 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.html	The default page for the sample. The Viewer's code gets loaded by this page.
sample-config.js	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
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the book reader Viewer.
modules folder	Contains uncompiled assets of the Viewer. These files will be compiled to viewer-

	in Gulpfile.js. The files in this folder are non-essential and do not need to be re-distributed.
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-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 assets. This file is non-essential and does not need to be re-distributed.
viewer-webtier folder	Contains files that implement the php layer of communication between the book reader 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.html	The default page for the sample. This page loads the Viewer.
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 bundle.js and bundle.css that are found in the viewer-assets folder.

## Folder contents: template-designer-sample

File / Folder	Description
viewer-assets folder	Contains the essential JavaScript, CSS, fonts, images, language data, and templates (HTML) that make up the book reader Viewer.
modules folder	Contains uncompiled assets of the Viewer. These files will be compiled to viewer-assets/js/bundle.js and viewer-assets/css/bundle.css by the build process defined in Gulpfile.js.  The files in this folder are non-essential and do not need to be re-distributed.
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-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 assets. This file is non-essential and does not need to be re-distributed.
viewer-webtier folder	Contains files that implement the php layer of communication between the book reader 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.html	The default page for the sample. This page loads the Viewer.
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 bundle.js and bundle.css that are found in the viewer-assets folder.

## Configuration with web.xml

The file **web.xml** is used to configure the connection settings between the web tier and PrizmDoc

documenting, but a little information about the configuration options is given below.

DocumentPath	The sample pulls named documents from this location. The DocumentPath must have read/write permissions in order for the file drag and drop functionality of the splash page to work.
PrizmApplicationServices[Scheme Host Port]	Specifies how to connect to the PrizmDoc Application Services

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>
```

## 3 - Integrate the Viewer with Your Application

- [Configure PAS in Your Server's Entry Point](#)
- [Create a Viewing Session](#)
- [Embed the Viewer](#)
- [Integrate Your Web Application with PAS](#)
  - [.NET WebForms](#)
  - [.NET MVC 5](#)
  - [PHP](#)
  - [JSP](#)

## Configure PAS in Your Server's Entry Point

How to configure Application Services in your server's entry point

Configuring the viewer

In the following examples, we will configure a route to reverse proxy all requests to Applications Services directly. While we do provide specific examples, there are ultimately many other ways in which you can

a web search for a reverse proxy on your specific stack should provide the necessary information.

In our examples, we will use **"pas-service"** as the example route. First, we will configure the viewer to use this route -- this step will be the same regardless of which server you use.

In the index page of your viewer (Default.aspx in C#, index.php in the PHP sample, etc.), find the configured `imageHandlerUrl` value and change it to **"pas-service"**. This will configure the viewer to send all of its requests to the "pas-service" endpoint. Note that all requests are sent relative to the index page, so you will have to make sure that the configured value is correct relative to the index page.

## Configure in IIS

First, make sure that the **URL Rewrite** and **Application Request Routing** modules are installed.

1. In the start menu, type **"Web Platform Installer"**. If this package manager is not installed, you can get it here: <http://www.microsoft.com/web/downloads/platform.aspx>
2. In the search box, type **"URL Rewrite"**.
3. Find the latest version (as of this writing, it is 2.0), and click the **"Add"** button.
4. Next, type **"Application Request Routing"** on the search box.
5. Find the latest version (as of this writing, it is 3.0), and click the **"Add"** button.
6. Click the **"Install"** button at the bottom and follow the prompts.

Add an empty folder somewhere in your project. This folder will become the redirect endpoint for the service. In this example, we will name this folder **"pas-service"** and place it in the root of the web application.

In the **Internet Information Services (IIS) Manager**, navigate to the "pas-service" folder in your web application, and select **"URL Rewrite"** from the options on the right. Click **"Add Rule(s)..."** in the sidebar to open the rewriting rules menu.

1. Select **"Reverse Proxy"** from the rule templates.
2. A menu will pop up if this is the first time you are using this feature, to prompt that proxy functionality must be enabled in AAR. Click the "OK" button to enable it.
3. We will assume that Application Services is running in its default configuration on the same machine as your application, under port 3000. If you are running it on a different machine or a different port, you will need to know these values here. In this example, we will use **"localhost:3000"**.
4. In the server name box, enter the location of Application Services. In our case, it is "localhost:3000".

This will result in the following web.config file in the empty "pas-service" folder:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <rewrite>
      <rules>
        <rule name="ReverseProxyInboundRule1" stopProcessing="true">
          <match url="(.*)" />
```

```
        </rules>
    </rewrite>
</system.webServer>
</configuration>
```

## Configure in Apache

You will need to make sure several modules are installed and enabled. These are: `mod_proxy`, `mod_proxy_http`, and `mod_rewrite`.

On a Debian based system, you can execute the following:

```
sudo a2enmod rewrite
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo service apache2 restart
```

On CentOS/RHEL, or other deployments of Apache, you will need to modify the Apache config, typically in `/etc/httpd/httpd.conf` or `/etc/apache2/apache2.conf`, to load the required modules. Make sure the following lines exist toward the top of the file (typically where all other modules are loaded):

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule rewrite_module modules/mod_rewrite.so
```

*Note: these lines may already be present, or be present and commented out.*

## Using the application conf file

Find your Apache conf file for your application. Typically, the default is set up to be `/etc/apache2/sites-enabled/000-default.conf` or `/etc/httpd/httpd.conf`, but this may vary for your application.

Determine the url that you would like to proxy to Application Services. In this example, we will assume that it is located at the root of the server and is named "**pas-service**" (e.g. <http://myserver/pas-service>).

Add the following to your application's VirtualHost:

```
ProxyPass /pas-service http://localhost:3000
```

Make sure to restart Apache when done.

## Using an .htaccess file

service. In this example, we will name this folder "**pas-service**" and place it in the root of the web application.

In order for `.htaccess` files to take effect, you may need to enable them. In the application config file, typically located at `/etc/httpd/httpd.conf` or `/etc/apache2/apache2.conf`, you will need to find the Directory declaration for your application. The default is `/var/www/`. In the directory block, make sure the `AllowOverride` line reads as such:

```
AllowOverride All
```

In the "pas-service" folder, add an `.htaccess` file. Add the following content to the file:

```
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^(.*)$ http://localhost:3000$1 [L,P]
</IfModule>
```

Make sure to restart Apache when done.

## Configure in NginX

Find your NginX conf file. Typically, this is located in `/etc/nginx/nginx.conf`, or a conf file inside `/etc/nginx/conf.d` or `/etc/nginx/default.d`, depending on your specific configuration.

Find the `server` block of the application in which you would like to deploy the viewer. In this example, we will assume the default server block.

Determine the url that you would like to proxy to Application Services. In this example, we will assume that it is located at the root of the server and is named "**pas-service**" (e.g. <http://myserver/pas-service>). Add the following code in your server block:

```
location /pas-service/ {
    rewrite ^/pas\-service(.*) $1 break;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass http://localhost:3000;
}
```

*Note: If you are using SELinux, you may need to run this command if proxying to an external server:*

```
sudo setsebool -P httpd_can_network_connect 1
```

## Create a Viewing Session

This section provides instructions on How to Use a Viewing Session in PrizmDoc Application Services:

- [Step 1: Create a Viewing Session](#)
- [Step 2 \(Optional\): Upload the Source Document](#)
- [Step 3: Request Content](#)

### Step 1: Create a Viewing Session

The recommended method for creating a Viewing Session in Application Services is through the [POST /ViewingSession](#) endpoint. It is also possible to create a Viewing Session directly through PrizmDoc Server, without using PAS. For the equivalent of this guide for that endpoint, see the [Using a Viewing Session](#) topic for more details.

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

```
POST http://localhost:3000/ViewingSession
Content-Type: application/json

{
  "source": {
    "type": "upload",
    "displayName": "sample.doc"
  },
  "render": {
    "html5": {
      "vectorTolerance" : 1.0
    }
  }
}
```

#### Example Response

```
200 OK
Content-Type: application/json

{
  "viewingSessionId": "-
gchUEYvBE50CgcWJajgoXcW7QD0I8zNDFlexD9hzbXkmrYlw8DrxJ-
KiHAf2oTAL_HiHK1MsstBlNgZFCrcJQ"
```

## Step 2 (Optional): Upload the Source Document

If a Viewing Session was created with a source type of "upload", a second request is required to provide it with the source document. For other source types, the source document will have already been specified through the `url`, `fileName`, or `documentId` properties, and this step should be skipped.

### Example Request

```
PUT http://localhost:3000/ViewingSession/u-
gchUEYvBE50CgcWJajgoXcW7QD0I8zNDFlexD9hzbXkmrYlw8DrxJ-
KiHAf2oTAL_HiHK1MsstBlNgZFCrcJQ/SourceFile
Content-Type: application/octet-stream
Accusoft-Secret: mysecretkey
{binary data}
```

### Example Response

```
200 OK
```

## Step 3: 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](#).

### Example Request

```
GET http://localhost:3000/Page/q/0?DocumentID=u-
gchUEYvBE50CgcWJajgoXcW7QD0I8zNDFlexD9hzbXkmrYlw8DrxJ-
KiHAf2oTAL_HiHK1MsstBlNgZFCrcJQ&ContentType=png
```

### Example Response

```
200 OK
Content-Type: image/png
{image data}
```

## Embed the Viewer

### Embed the Viewer

The PrizmDoc Viewer has been designed to be integrated into both new and existing web applications. Using the familiar jQuery plugin syntax, you can embed the Viewer into your application, and easily configure it to meet your needs.

*The Viewer requires PrizmDoc Server (either hosted by Accusoft or self-hosted) in order to supplement viewing session requests. The Viewer also needs access to PrizmDoc Application Services; for more information on setting up PrizmDoc Application Services routing, see [Configure PrizmDoc Application](#)*

## Setting up your Application to use the Viewer

Before initializing the Viewer using the jQuery plugin, some basic setup is required.

### 1. Add `<meta>` tags to the head of your page

These `<meta>` tags are important for utilizing the responsive features of the Viewer across devices and enabling support for Internet Explorer. Include these in the in the `<head>` of your page, if they are not already there.

#### Example

```
<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" />
```

### 2. Include the CSS files

The Viewer has a dependency on normalize, which needs to be loaded before all other styles. Then, load the two stylesheets that are part of the Viewer: `viewercontrol.css`, which styles the elements from the underlying viewer API, and `viewer.css`, which styles the viewer itself.

To use the Viewer's user interface chrome, link to these stylesheets in the `<head>` of your page:

#### Example

```
<link rel="stylesheet" href="css/normalize.min.css">
<link rel="stylesheet" href="css/viewercontrol.css">
<link rel="stylesheet" href="css/viewer.css">
```

These files can be found in the `viewer/css` directory under the Prizm install directory.

### 3. Include the JavaScript files

The Viewer has a dependency on jQuery and jQuery Hotkeys, as well as Underscore. These need to be loaded first, followed by `viewercontrol.js`, the underlying viewing API, and `viewer.js`, the viewer code itself. Both `viewercontrol.js` and `viewer.js` are updated with each release of PrizmDoc.

#### Example

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">
</script>
<script src="js/jquery.hotkeys.min.js"></script>
<script src="js/underscore.min.js"></script>
<script src="js/viewercontrol.js"></script>
<script src="js/viewer.js"></script>
```

These files can be found in the `viewer/js` directory under the Prizm install directory. The order the scripts are loaded is important, and should be added in the order shown here.

#### 4. Add a `<div>` to hold the Viewer

To add the Viewer to your application, you will need to create a `<div>` element with a unique ID where you would like to place the Viewer. In this example, we are using `myDiv` as our unique ID, but you may name it whatever fits your application's coding guidelines.

#### Example

```
<div id="myDiv"></div>
```

#### 5. Load Additional Resources and Initialize the Viewer

The Viewer ships with several additional resources which are required by the viewer at runtime. These resources can all be found in the `prizm` installation directory under the `viewer` subdirectory.

#### languages

The viewer isolates its language resources in a separate file under the `languages` subdirectory. The language resources are contained in a json file, `en-US.json`, and are required by the viewer at initialization. This file should be loaded at runtime, allowing for a regionalized viewing experience, and provided during initialization in the `pluginOptions` object, using the `language` key.

#### templates

Much of the viewer's layout is defined in `html` template files. These files are found under the `templates` subdirectory and are required by the viewer at initialization. The templates correspond to HTML files available in all samples -- the filenames use the same name as the template names here, appending "Template.html" to the end -- for example, the template `fileName` would correspond to a `fileNameTemplate.html` file. These files should be loaded at runtime and provided during initialization in the `pluginOptions` object, using the `template` key.

#### img

Several images containing icons and other graphic elements are required by the viewer. These can be found under the `img` subdirectory

#### Directory Structure

The steps in this example assume the following layout for the resource directories.

```
|— index.html
|— css
|— img
|— js
|— languages
└— templates
```

Now that the setup is complete, you are ready to embed the full-featured, responsive Viewer into your web application using jQuery's plugin syntax.

The Viewer's user interface and behavior can be configured when it is embedded using JavaScript parameters. Only four parameters are required to initialize the Viewer: the previously described language, and template as well as the documentID and imageHandlerUrl which are described below. See the [jQuery viewer plugin documentation](#) for more information about each of these parameters.

## documentID

This is the viewing session id returned upon the creation of a new viewing session. The id is used by the viewer to make requests for document artifacts and to perform other operations.

## imageHandlerUrl

This is the url of the Prizm Application Services reverse proxy (or your own custom web tier). The viewer will make requests through this endpoint. See [Configure PrizmDoc Application Services in your Server's Entry Point](#) for examples of setting up the reverse proxy.

Add the following snippet to your JavaScript file to set the required properties:

## Example

```
var pluginOptions = {
  documentID: viewingSessionId,
  imageHandlerUrl: '/pas-service',
  language: languageItems,
  template: htmlTemplates
}

$('#myDiv').pccViewer(pluginOptions);
```

If you chose a different ID than the example in Step 4 above, replace #myDiv with your chosen ID value.

There are lots of other configuration options for the Viewer. For detailed information on all the configuration options available to you, see the [Configuration Options](#) section.

## Troubleshooting

Here are a few issues that may arise while initializing the Viewer that 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.

```
Cannot read property 'viewer' of undefined
```

This error is thrown when viewer.js cannot find the template object. Make sure that the template object exists and that there is a viewer property.

## 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.

If you're still having issues setting up the Viewer, visit our [Support](#) page.

## See Also

- [Design Basics](#)
- [Architecture & Design](#)
- [Set up Viewer Samples \(Optional\)](#)
- [Configure the Viewer](#)
- [Configure PrizmDoc Application Services in your Server's Entry Point](#)
- [Customize the Viewer](#)
- [Developer Reference](#)

## Integrate Your Web Application with PAS

This section contains information for integrating your Web Application with PAS based on the language you are using:

- [.NET WebForms](#)
- [.NET MVC 5](#)
- [PHP](#)
- [JSP](#)

 This is for evaluation only. In a production environment, follow the instructions in [Configuring PAS in Your Server's Entry Point](#).

## .NET WebForms

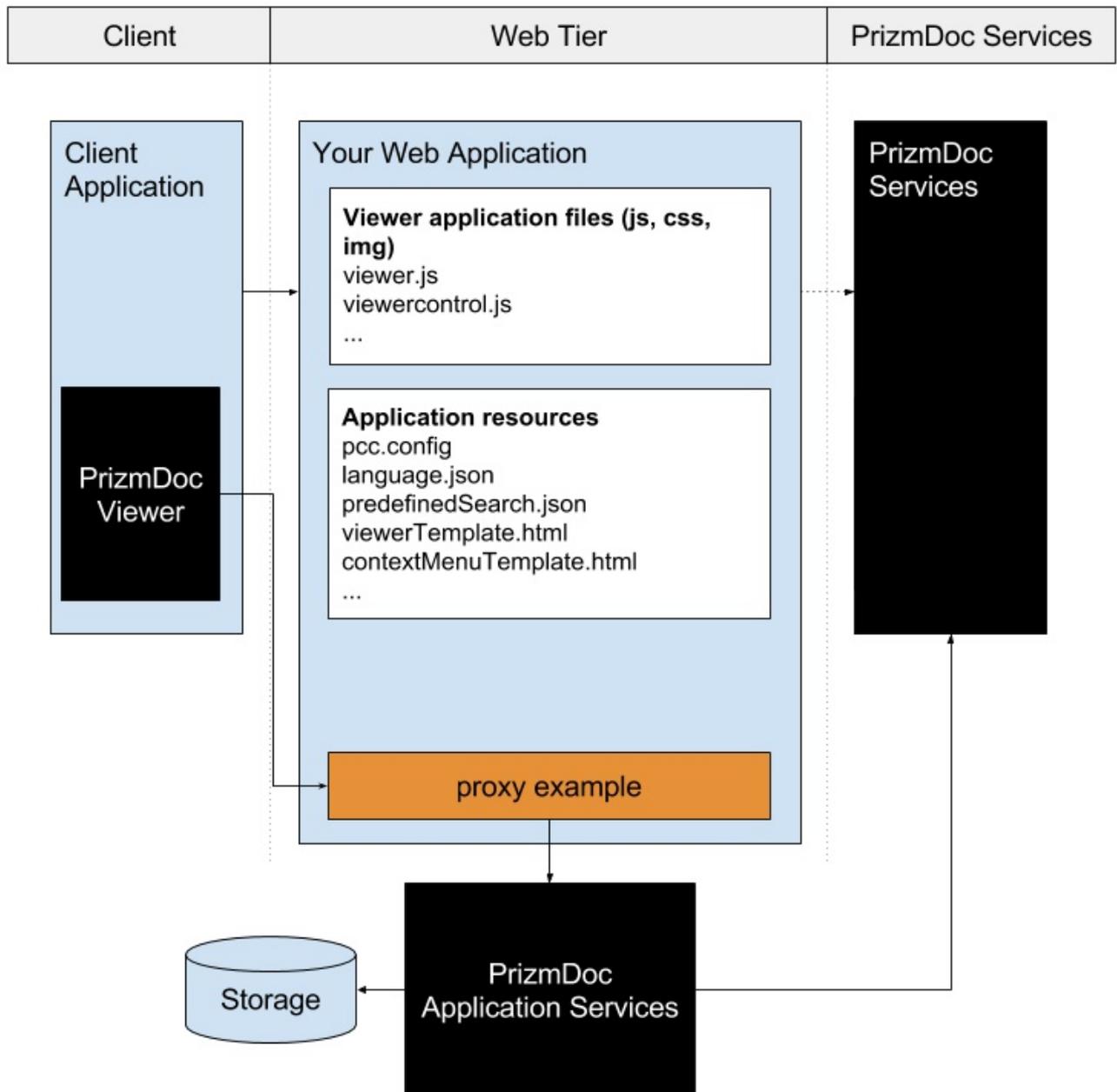


Figure 1: .NET WebForms Sample Setup

## First Steps

Before integrating with your application environment, a helpful first step is to install the provided C# sample. With a working sample, you will have a proof-of-concept that demonstrates the functionality of the PrizmDoc Server working together with your web server. To get started with setting up a C# sample, refer to [How to Configure C# Samples](#).

Once installed, the sample will load in the following order (refer to Figure 1):

1. **Default.aspx** is called on the web server by a request initiated by a web browser.
2. **Default.aspx** executes and bundles together various application resources like user interface templates,

3. Once the HTML page generated by **Default.aspx** is loaded in to the web browser, the Viewer application files (javascript, css, and image resources) are then loaded from the web server.
4. Upon receiving the files in the web browser, the Viewer application starts and requests a document for display from the web server. By default, this request is routed through **pcc.ashx**, as are all other subsequent service requests. **pcc.ashx** acts as a proxy between the Viewer and PrizmDoc Application Services (PAS). Note that this can be configured to work directly through IIS instead of using application logic. Refer to the [Configure PrizmDoc Application Services in Your Server's Entry Point](#) 'how-to' topic.
5. At this stage, the Viewer is fully loaded and displaying a document. It will now respond to user interaction by processing it locally in the browser or, if not possible, by sending service requests to the PrizmDoc Application Services (PAS).

## Integration

### The Front End

The Viewer loads like most modern web applications: a server script is called, which then presents an HTML page to the browser. The browser loads the page's linked JavaScript and CSS files, which triggers the front-end web application to initiate. Because the technologies are standard web ones, you can take the js, img, and css folders provided by the sample and plug them into your existing web platform. The sample places the JavaScript and CSS folders at the same directory level as the application launching **Default.aspx**. However, the location of these resources can be changed by following the directions in [Using a Custom Resource Path](#).

Also, for details on embedding the Viewer on existing pages, see [Embedding the Viewer](#).

### The Back End

An important point to make is that while your web application, PAS and the PrizmDoc Server can be installed on the same machine, they do not have to be. Because the web server communicates with the other services over RESTful network calls, they can be located anywhere where they can all interact with each other over HTTP. In the sample, the PAS host URL and port are defined in the pcc.config file with several parameters prefixed with **PrizmApplicationServices**:

Figure 2: The pcc.config file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Config>
3    <PrizmApplicationServicesScheme>http</PrizmApplicationServicesScheme>
4    <PrizmApplicationServicesHost>localhost</PrizmApplicationServicesHost>
5    <PrizmApplicationServicesPort>3000</PrizmApplicationServicesPort>
6  </Config>
7
```

In the default installation, the web tier sample is configured with the default location of the PrizmDoc Application Services (PAS). When deploying to multiple servers, you may need to change the values here to allow the web tier sample to communicate with PAS.

Another important file is **pcc.ashx**. This acts as a router between the front end Viewer and PAS. So **pcc.ashx** must be able to receive and respond to requests from the Viewer. The sample places **pcc.ashx** in the viewer-webtier directory and informs the Viewer of its location via the **imageHandlerUrl** parameter in **Default.aspx**:

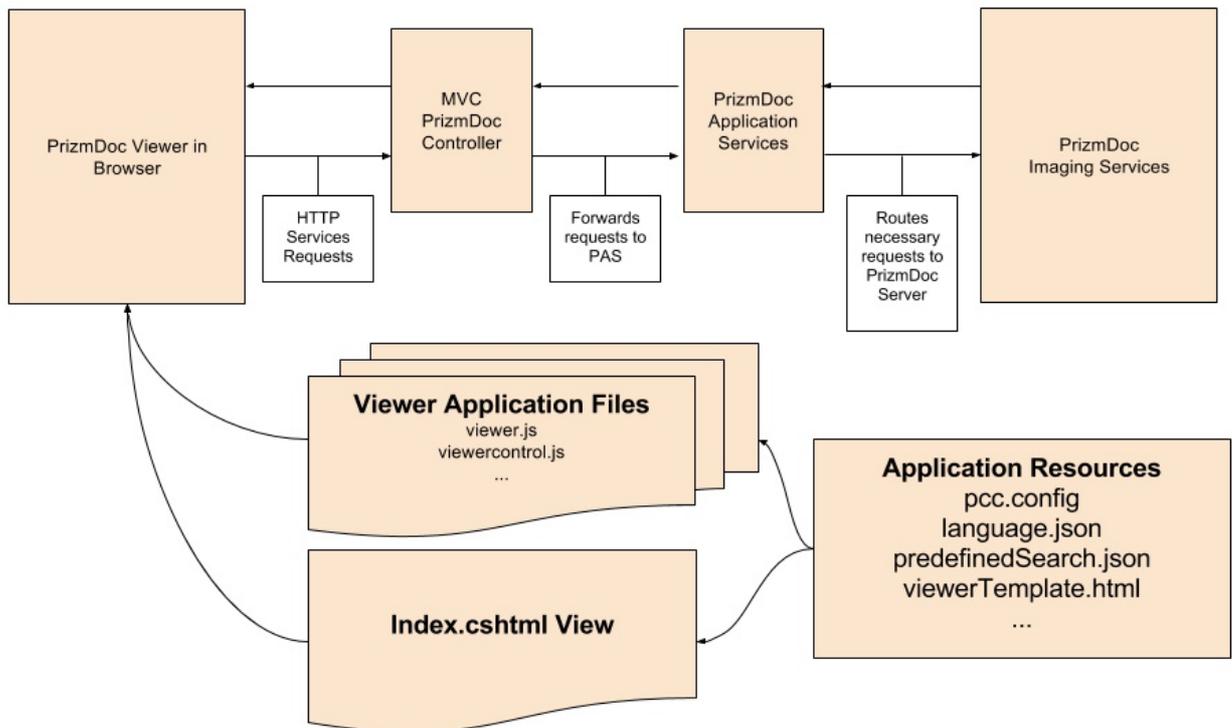
Figure 3: Example of imageHandlerUrl in plugin options

```
2  documentID: viewingSessionId,
3  language: languageItems,
4  annotationsMode: "LayeredAnnotations",
5  documentDisplayName: originalDocumentName,
6  imageHandlerUrl: "viewer-webtier/pcc.ashx",
7  immediateActionMenuMode: "hover",
8  predefinedSearch: searchTerms,
9  template: htmlTemplates,
10 redactionReasons: redactionReasons,
11 signatureCategories: "Signature,Initials,Title",
12 resourcePath: "viewer-assets/img"
13 };
14
15 $(document).ready(function () {
16     var viewerControl = $("#viewer1").pccViewer(pluginOptions).viewerControl;
17 });
```

So, you can easily change the location of **pcc.ashx** and then adjust the **imageHandlerUrl** parameter to match. Just be aware that **pcc.ashx** works together with the classes found in the **App\_Code** folder. Those files will need to be moved with **pcc.ashx**.

## .NET MVC 5

Before you get started, it's helpful to see an application overview of the default .NET MVC 5 sample setup:



## First Steps

Before integrating with your application environment, a helpful first step is to install the provided MVC 5 sample. With a working sample, you will have a proof-of-concept that demonstrates the functionality of the PrizmDoc Server working together with your web server. To get started with setting up an MVC 5 sample, refer to [How to Configure the .NET MVC 5 Sample](#).

Once installed, the sample will load in the following order (refer to Figure 1 above):

1. The view (Views/{Viewer}/Index.cshtml) is called on the web server by a request initiated by a web browser.
2. The requested view and its controller will then execute and bundle together various application resources like user interface templates, language files, and predefined search queries.
3. Once the HTML page generated by the view is loaded into the web browser, the Viewer application files (javascript, css, and image resources) are then loaded from the web server.
4. Upon receiving the files in the web browser, the Viewer application starts and requests a document for display from the web server. This request is routed through PccController, as are all other subsequent service requests. The PccController acts as a router between the Viewer app and the PrizmDoc Application Services. Going back to our example, PccController will request the document from PrizmDoc Application Services and will send the response back to the Viewer.
5. At this stage, the Viewer is fully loaded and displaying a document. It will now respond to user interaction by processing it locally in the browser or, if not possible, by sending service requests to the web server, which can then forward them on to PrizmDoc Application Services.

## Integration

The following example shows the .NET MVC 5 sample directory structure:

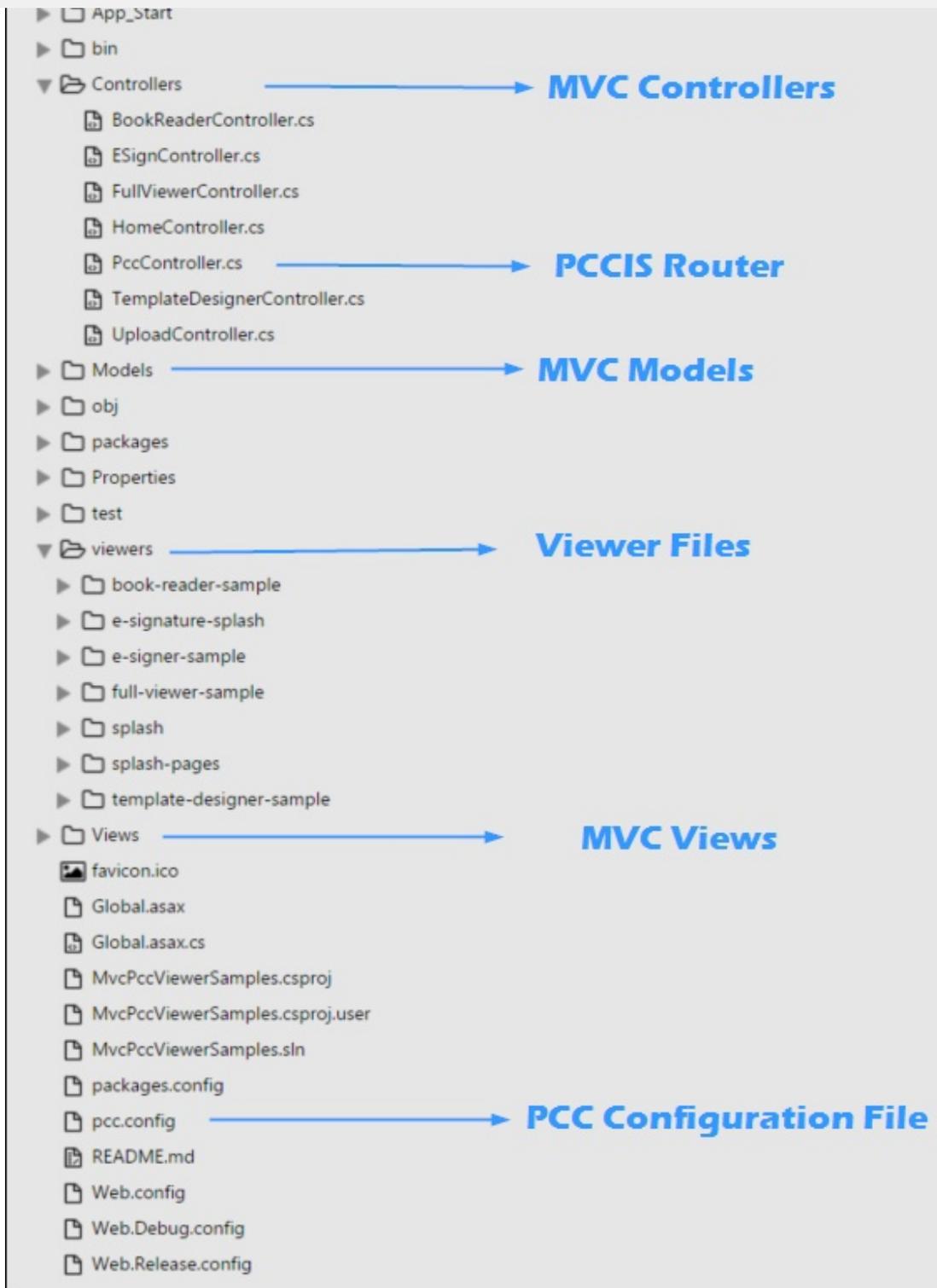


Figure 2: .NET MVC 5 Sample Directory Structure

## The Front End

As depicted in Figure 1, the Viewer loads like most modern web applications: a server script is called, which then presents an HTML page to the browser. The browser loads the page's linked JavaScript and CSS files, which triggers the front-end web application to initiate. Because the technologies are standard web ones, you

sample places the javascript and CSS folders in the viewers folder. However, the location of these resources can be changed by following the directions in [Using a Custom Resource Path](#). Also, for details on embedding the Viewer on existing pages, see [Embedding the Viewer](#).

## The Back End

An important point to make is that while the web server and PrizmDoc Application Services (PAS) can be installed on the same machine, it is not a requirement because the web server communicates with PrizmDoc Application Services over HTTP calls; PrizmDoc Application Services can be located anywhere that the web server can reach it via that URL. In the sample, the URL is defined in the pcc.config file with several parameters prefixed with PrizmApplicationServices:

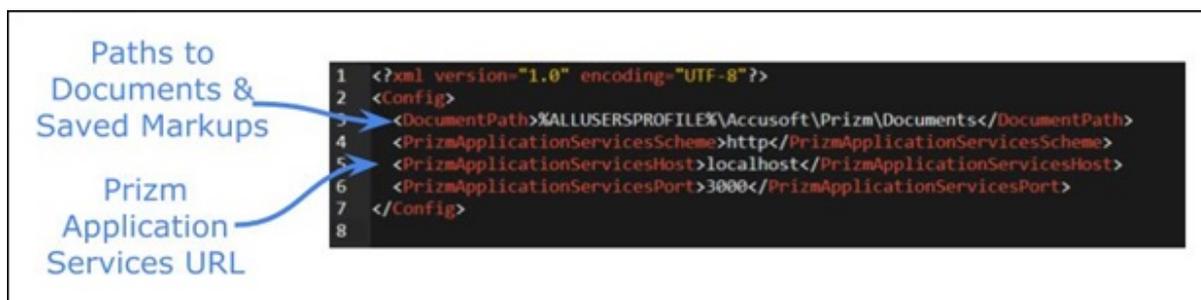


Figure 3: Excerpt from pcc.config

Also in that file there is one important setting that defines the document location (DocumentPath). As implemented in the .NET MVC 5 sample, the document path is the directory on the web server that holds the original document in its native format. It's highly likely that these parameters will need to be updated for your server environment, thus they should always be reviewed after installing the sample. Loading a document from the file system into PrizmDoc Application Services is just one approach. As an example, the document could alternatively originate from a database or URL. However, to keep the sample straightforward, the original document resides on the local file system.

Another important file is the PccController. This acts as a router between the front end Viewer and the PrizmDoc Application Services. So PccController must be able to receive and respond to requests from the Viewer. The sample routes requests to PccController via a route that you may find in App\_Start/RouteConfig.cs. the Viewer is configured to point its requests to this location via the imageHandlerUrl parameter which is located in the View that was requested:

```
viewerAssetsPath: "viewers/full-viewer-sample/viewer-assets",
languageFile: "en-US.json",
createSessionUrl: "pcc/CreateSession",
imageHandlerUrl: "pcc"
};

var pluginOptions = {
  documentID: viewingSessionId,
  language: languageItems,

  annotationsMode: "LayeredAnnotations",
  documentDisplayName: originalDocumentName,
  imageHandlerUrl: sampleConfig.imageHandlerUrl,
  immediateActionMenuMode: "hover",
  predefinedSearch: searchTerms,
  template: htmlTemplates,
  redactionReasons: redactionReasons,
  signatureCategories: "Signature,Initials,Title",
  resourcePath: sampleConfig.viewerAssetsPath + "/img",
  uiElements: {
    download: true,
    fullScreenOnInit: true,
    advancedSearch: true
  }
};
```

Figure 4: Example of imageHandlerUrl in plugin options

You can easily change the location of PccController route and then adjust the imageHandlerUrl parameter to match.

## PHP

Before you get started, it's helpful to see an application overview of the default PHP sample setup:

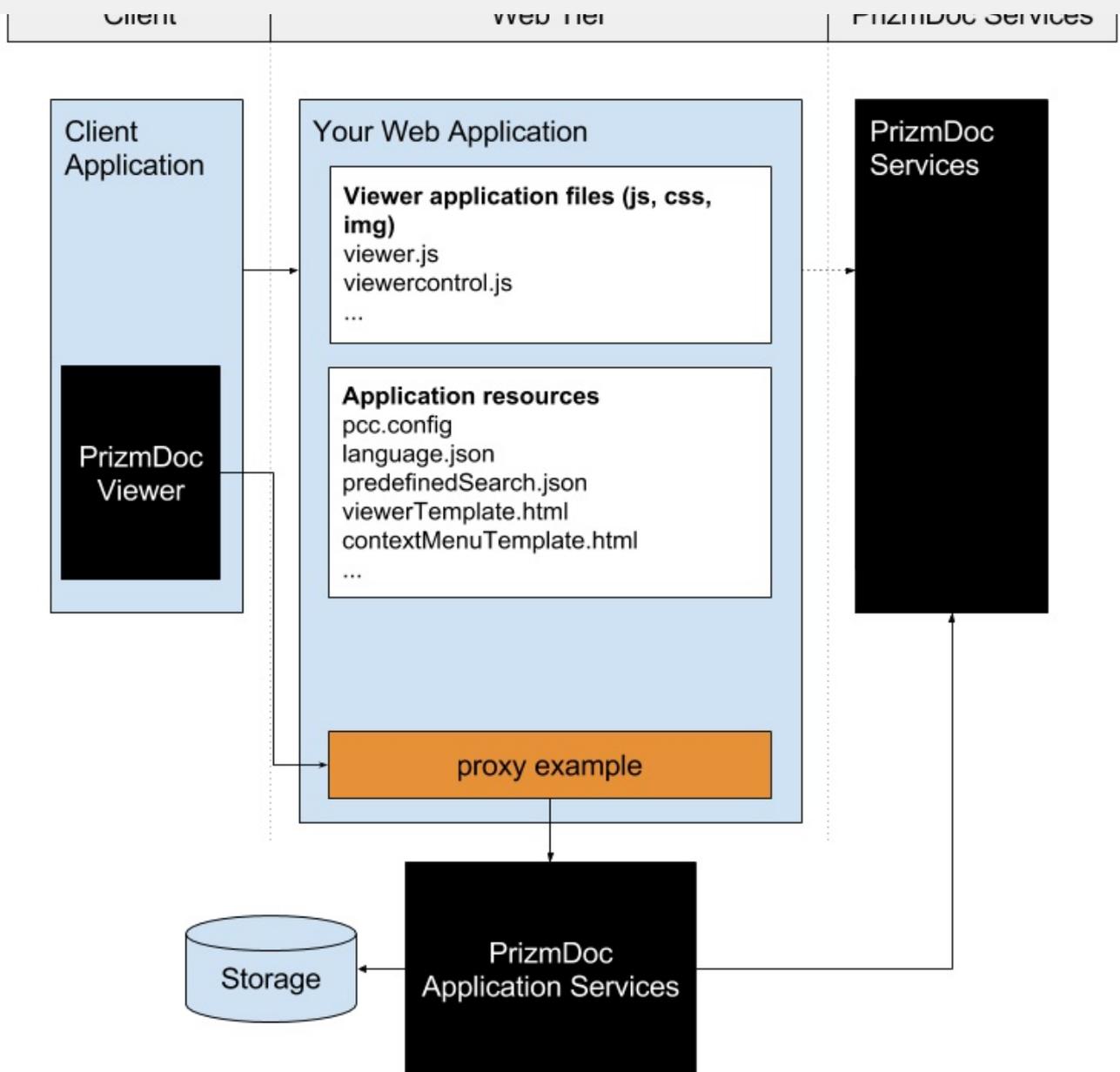


Figure 1: PHP Sample Setup

## First Steps

Before integrating with your application environment, a helpful first step is to install one of the provided samples. With a working sample, you will have a proof-of-concept that demonstrates the functionality of the PrizmDoc Server working together with your web server. To get started with setting up a sample, refer to [How to Configure php Samples](#).

Once installed, the sample will load in the following manner (refer to Figure 1 above):

1. **index.php** is called on the web server by a request initiated by a web browser.
2. **index.php** executes and bundles together various application resources like user interface templates, language files, and predefined search queries.

resources) are then loaded from the web server.

4. Upon receiving the files in the web browser, the Viewer application starts and requests a document for display from the web server. This request is routed through **pas.php** as are all other subsequent service requests. **pas.php** acts as a router between the Viewer and PrizmDoc Application Services (PAS). Note that this can be configured to work directly through IIS instead of using application logic. Refer to the [Configure PrizmDoc Application Services in Your Server's Entry Point](#) 'how-to' topic.
5. At this stage, the Viewer is fully loaded and displaying a document. It will now respond to user interaction by processing it locally in the browser or, if not possible, by sending service requests to the PrizmDoc Application Services (PAS).

## Integration

### The Front End

The Viewer loads like most modern web applications: a server script is called, which then presents an HTML page to the browser. The browser loads the linked JavaScript and CSS files, which triggers the front-end web application to initiate. Because the technologies are standard web ones, you can take the js, img, and css folders provided by the sample and plug them in to your existing web platform. The sample places the JavaScript and CSS folders at the same directory level as the application launching **index.php**. However, the location of these resources can be changed by following the directions in [Using a Custom Resource Path](#).

Also, for details on embedding the Viewer on existing pages, see [Embedding the Viewer](#).

### The Back End

An important point to make is that while your web application, PAS, and the PrizmDoc Server can be installed on the same machine, they do not have to be. Because the web server communicates with the other services over RESTful network calls, they can be located anywhere where they can all interact with each other over HTTP. In the sample, the PAS host URL and port are defined in the **pcc.config** file with several parameters prefixed with **PrizmApplicationServices**.

Figure 2: The pcc.config file

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Config>
3   <PrizmApplicationServicesScheme>http</PrizmApplicationServicesScheme>
4   <PrizmApplicationServicesHost>localhost</PrizmApplicationServicesHost>
5   <PrizmApplicationServicesPort>3000</PrizmApplicationServicesPort>
6 </Config>
7
```

In the default installation, the web tier sample is configured with the default location of the PrizmDoc Application Services (PAS). When deploying to multiple servers, you may need to change the values here to allow the web tier sample to communicate with PAS.

Another important file is **pas.php**. This acts as a router between the front end Viewer and PrizmDoc Application Services (PAS). So **pas.php** must be able to receive and respond to requests from the Viewer. The sample places **pas.php** in the viewer-webtier directory and informs the Viewer of its location via the **imageHandlerUrl** parameter in **index.php**:

Figure 3: Example of imageHandlerUrl in plugin options

```
2     documentID: viewingSessionId,  
3     language: languageItems,  
4     annotationsMode: "LayeredAnnotations",  
5     documentDisplayName: originalDocumentName,  
6     imageHandlerUrl: "viewer-webtier/pas.php",  
7     immediateActionMenuMode: "hover",  
8     predefinedSearch: searchTerms,  
9     template: htmlTemplates,  
10    redactionReasons: redactionReasons,  
11    signatureCategories: "Signature,Initials,Title",  
12    resourcePath: "viewer-assets/img"  
13 };  
14  
15 ▾ $(document).ready(function () {  
16     var viewerControl = $("#viewer1").pccViewer(pluginOptions).viewerControl;  
17 });
```

So, you can easily change the location of **pas.php** and then adjust the **imageHandlerUrl** parameter to match. Just be aware that **pas.php** works together with the rest of the files in the viewer-webtier directory. Those files will need to be moved with **pas.php**.

## JSP

Before you get started, it's helpful to see an application overview of the default JSP sample setup:

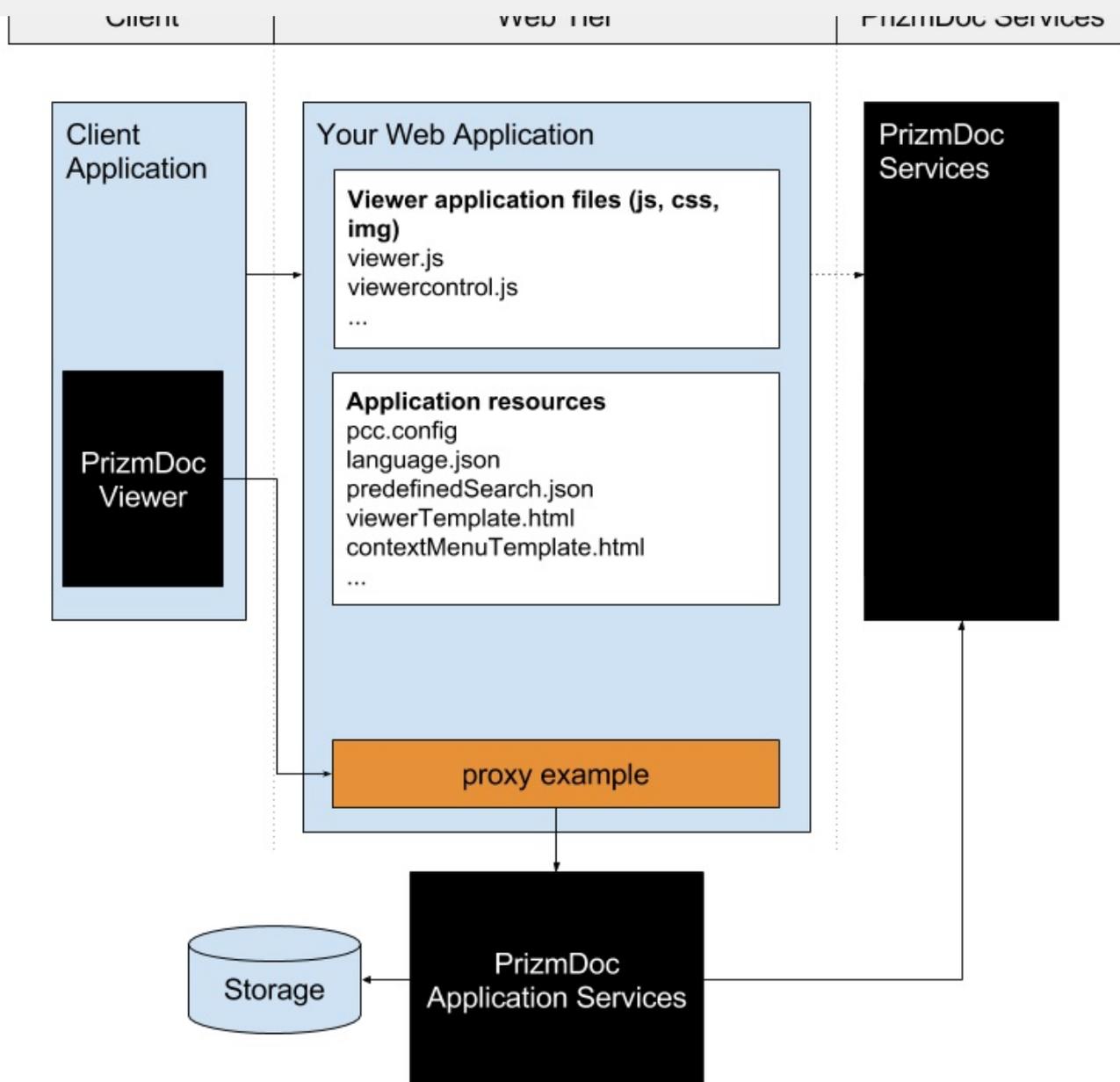


Figure 1: JSP Sample Setup

## First Steps

Before integrating with your application environment, a helpful first step is to install the provided JSP sample. With a working sample, you will have a proof-of-concept that demonstrates the functionality of the PrizmDoc Server working together with your web server. To get started with setting up a JSP sample, refer to [How to Configure JSP Samples](#).

**⚠ Note that JDK 1.7 and JRE 1.7+ are required.**

Once installed the sample will load in the following order (refer to Figure 1 above):

1. **index.jsp** is called on the web server by a request initiated by a web browser.

language files, and predefined search queries.

3. Once the HTML page generated by **index.jsp** is loaded in to the web browser, the Viewer application files (javascript, css and image resources) are then loaded from the web server.
4. Upon receiving the files in the web browser, the Viewer application starts and requests a document for display from the web server. This request is routed through **pas.jsp** as are all other subsequent service requests. **pas.jsp** acts as a proxy between the Viewer and PrizmDoc Application Services (PAS). Note that this can be configured to work directly through IIS instead of using application logic. Refer to the [Configure PrizmDoc Application Services in Your Server's Entry Point](#) 'how-to' topic.
5. At this stage, the Viewer is fully loaded and displaying a document. It will now respond to user interaction by processing it locally in the browser or, if not possible, by sending service requests to PrizmDoc Application Services (PAS).

## Integration

### The Front End

The Viewer loads like most modern web applications: a server script is called which then presents an HTML page to the browser. The browser loads the page's linked JavaScript and CSS files, which triggers the front-end web application to initiate. Because the technologies are standard web ones, you can take the js, img, and css folders provided by the sample and plug them into your existing web platform. The sample places the JavaScript and CSS folders at the same directory level as the application launching **index.jsp**. However, the location of these resources can be changed by following the directions in [Using a Custom Resource Path](#).

Also, for details on embedding the Viewer on existing pages, refer to [Embedding the Viewer](#).

### The Back End

An important point to make is that while your web application, PAS, and the PrizmDoc Server can be installed on the same machine, they do not have to be. Because the web server communicates with the other services over RESTful network calls, they can be located anywhere where they can all interact with each other over HTTP. In the sample, the PAS host URL and port are defined in the "**WEB-INF\web.xml**" configuration file with several parameters prefixed with **PrizmApplicationServices**:

Figure 2. The web.xml file

```
2 <web-app version="2.5"
3     xmlns="http://java.sun.com/xml/ns/javaee"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
6 <description>Prizm Content Connect Sample</description>
7 <display-name>Prizm Content Connect Sample</display-name>
8 <servlet>
9     <servlet-name>pas.jsp</servlet-name>
10    <jsp-file>/viewer-webtier/pas.jsp</jsp-file>
11    <load-on-startup>0</load-on-startup>
12 </servlet>
13 <servlet-mapping>
14     <servlet-name>pas.jsp</servlet-name>
15     <url-pattern>/pas.jsp/*</url-pattern>
16 </servlet-mapping>
17 <context-param>
18     <description>Prizm Application Services Scheme</description>
19     <param-name>PrizmApplicationServicesScheme</param-name>
20     <param-value>http</param-value>
21 </context-param>
22 <context-param>
23     <description>Prizm Application Services Host name</description>
24     <param-name>PrizmApplicationServicesHost</param-name>
25     <param-value>localhost</param-value>
26 </context-param>
27 <context-param>
28     <description>Prizm Application Services Port</description>
29     <param-name>PrizmApplicationServicesPort</param-name>
30     <param-value>3000</param-value>
31 </context-param>
32 </web-app>
33
```

In the default installation, the web tier sample is configured with the default location of the PrizmDoc Application Services (PAS). When deploying to multiple servers, you may need to change the values here to allow the web tier sample to communicate with PAS.

Another important file is **pcc.jsp**. This acts as a router between the front end Viewer and the PrizmDoc Server. So **pcc.jsp** must be able to receive and respond to requests from the Viewer. The sample places **pcc.jsp** one directory above the application launching **index.jsp** and informs the Viewer of its location via the **imageHandlerUrl** parameter in **index.jsp**:

Figure 3. Example of imageHandlerUrl in plugin options

```
1 var pluginOptions = {
2     documentID: viewingSessionId,
3     language: languageItems,
4     annotationsMode: "LayeredAnnotations",
5     documentDisplayName: originalDocumentName,
6     imageHandlerUrl: "../pas.jsp",
7     immediateActionMenuMode: "hover",
8     predefinedSearch: searchTerms,
9     template: htmlTemplates,
10    redactionReasons: redactionReasons,
11    signatureCategories: "Signature,Initials,Title",
12    resourcePath: "viewer-assets/img"
13 };
14
15 $(document).ready(function () {
16     var viewerControl = $("#viewer1").pccViewer(pluginOptions).viewerControl;
17 });
```

So, you can easily change the location of the **pas.jsp** servlet and then adjust the **imageHandlerUrl** parameter to match. Just be aware that **pas.jsp** servlet works together with the PrizmApplicationServices class found in the **src** folder.

## Administrator Guide

This section contains the following information:

- Security Guidance
- Sizing Servers
  - PrizmDoc Server
  - PrizmDoc Application Services
- Licensing
  - Evaluation Licensing
    - Command-Line Mode
    - Troubleshoot Evaluation Licensing
      - No Internet Connection
      - Exceeded Installation Limit
      - License Expired
      - System is Already Licensed for Evaluation
  - Deployment Licensing
    - Node-Locked
      - Without an Internet Connection
    - OEM
    - Cloud
    - Command-Line Mode
  - Feature Licensing
- System Configuration
  - Overview
  - Configure the Viewer
    - Configuration Options
      - Initialization Parameters
      - uiElements
      - Pre-Loaded Search Parameters
      - Configuring Skinny Comments Panel
      - Defining the View Mode
      - Digital Rights Management Configuration
      - Enabling Content Encryption
      - How to use Predefined Search
      - Localizing the Viewer
      - Using a Custom Resource Path
      - Adding Custom Image Stamps
  - Configure PrizmDoc Application Services (PAS)
    - PrizmDoc Application Services (PAS) Configuration
    - PrizmDoc Application Services Database Administration & Maintenance

- Configure the PrizmDoc Server
  - Central Configuration
  - Upgrade to Central Configuration
  - Configuring Ports
  - Format Detection Configuration & Use
  - Adjust Caching Parameters for PrizmDoc Server
  - Adjust Office Conversion Settings for Optimal Performance
  - Adjust Vector Conversion Settings for Optimal Performance
  - Change Encryption Keys for Public use Token Generation
  - Configure Image Frame Rendering in the PDF Conversion Service
  - Configure Log File Locations
  - Configure Microsoft Office Conversion Connectivity
  - Customize Excel Pagination Settings for PrizmDoc Server
  - Customize Excel Document View Settings for PrizmDoc Server
  - Customize Text File Encoding for PrizmDoc Server
  - Disable Excel Pagination for PrizmDoc Server
  - Implement PrizmDoc Server Caching Strategies
  - Substitute Fonts for Office Rendering Fidelity
- Start & Stop PrizmDoc Application Services
  - Linux
  - Windows
- Start & Stop PrizmDoc Server
  - Linux
  - Windows
- Multi-Server Environments
  - PrizmDoc Server
    - Optimize Cache Performance for Multi-Server Mode
    - Affinity Tokens & Multi-Server Mode
    - Deprecated Configuration Properties
  - PrizmDoc Application Services (PAS)
    - Optimize Cache Performance for Multi-Server Environments
    - Run PrizmDoc Application Services on Multiple Servers
- Troubleshoot
  - Windows
  - Linux
  - Check the System's Health
  - Connection Issues
  - Error Reporting
    - Getting Started
    - Accusoft Policy on Log Changes
    - Search Tips

- [Form Field Detector Error Messages](#)

## Security Guidance

The following sections discuss items that should be considered before deploying your application using PrizmDoc.

- [PrizmDoc Server](#)
- [PrizmDoc Server Administration](#)
- [Ports](#)
- [Secure Viewing Sessions](#)

 To prevent attacks on viewing sessions, refer to the "Secure Viewing Sessions" section below.

### PrizmDoc Server

PrizmDoc Server is designed to be run as an internal web service. Steps should be taken to ensure that PrizmDoc Server is not accessible to end-users or the public internet. Typically, this would involve configuring a firewall in-front of PrizmDoc Server to block access to the port it is using. See the "Ports" section below for specific port information about PrizmDoc Server.

### PrizmDoc Server Administration

PrizmDoc Server includes an 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 administration API and demonstrates potential use cases.

The administration API provides information that can be helpful in diagnosing problems, but which may also be considered sensitive, like document information and specific processing tasks. Because of this, the administration sample or any application accessing the administration API of PrizmDoc Server should not be accessible to end-users or the public internet.

### Ports

The following are the default ports that should be open to access PrizmDoc Server.

#### Single-server Mode:

- **18681** – PrizmDoc Server Entry Point (SEP) default port

#### Multi-server Mode:

- **18681** – PrizmDoc Server Cloud Entry Point (CEP) default port
- **18682** – PrizmDoc Server Entry Point (SEP) default port

 PrizmDoc Server also uses a number of ports for internal purposes. These ports must not be accessible from outside of the server. See [Configuring Ports](#) for more details.

### Secure Viewing Sessions

The central configuration file contains properties that can help prevent users from setting inappropriate values should they 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.

For more information 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:

#### Example - Central Configuration Properties

```
#
# 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]
```

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

The **pcc.config** file contains element tags that can help prevent users from setting inappropriate values should they 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 following tags put limits on properties sensitive to abusive attacks:

## Tags

```
<!--
  The regular expression check on ViewingSessionProperties.externalId to ensure appropriate values
  are being set. The default is to allow any string values.
-->
<ViewingSessionPropertyExternalId>.*</ViewingSessionPropertyExternalId>

<!--
```

```
-->
<ViewingSessionPropertyDocumentExtension>.*</ViewingSessionPropertyDocumentExtension>

<!--
  The minimum and maximum values allowed for ViewingSessionProperties.countOfInitialPages. Value
  of 0 means do all pages if min set to zero. The max value can be zero or a maximum value allowed
  for this property setting.
-->

<ViewingSessionPropertyCountOfInitialPages>min=0,max=10</ViewingSessionPropertyCountOfInitialPages>

<!--
  The minimum and maximum dpi values allowed for rendering images.
-->
<Html5RenderRasterResolution>min=100,max=300</Html5RenderRasterResolution>

<!--
  The permitted values for alwaysUseRaster can be true, false, or any (which means don't care).
  The default here is false which means svg files can be rendered.
-->
<Html5RenderAcceptableRasterValue>>false</Html5RenderAcceptableRasterValue>

<!--
  The permitted values for serverCaching which can be none, full or any (which means take whatever
  is set). The default is none.
-->
<ViewingSessionPropertyServerCaching>none</ViewingSessionPropertyServerCaching>
```

## Sizing Servers

This section contains the following information:

- [PrizmDoc Server](#)
- [PrizmDoc Application Services](#)

## PrizmDoc Server

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 at [support@accusoft.com](mailto:support@accusoft.com).

- [How Content Can Affect These Recommendations](#)
- [How Hardware Can Affect System Performance](#)
- [Example Server Configurations](#)
- [Additional Considerations for Pre-Conversion Services](#)

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 file 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.

each new request. We often see customers proposing hardware configurations that possess multiple core and thread 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

Through our experimentation and testing, we recommend that a server contain 2GB of Memory per Physical CPU core for use by PrizmDoc Server for processing. Any Memory considerations for a RAM Disk or Shared Memory Volume should be made in addition to this recommendation.

## 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 few hundred pages.

### Minimum Recommendation

- Windows: 5 per minute
- Linux: 10 per minute

### Hardware Details:

- Processors Cores: 4
- Memory: 32GB
- Hard Drive Type: SSD
- AWS: r3.Xlarge

### Moderate Recommendation

- Linux: 18 per minute

#### Hardware Details:

- Processors Cores: 8
- Memory: 64GB
- Hard Drive Type: SSD
- AWS: r3.2xlarge

#### High-End Recommendation

- Windows: 15 per minute
- Linux: 30 per minute

#### Hardware Details:

- Processors Cores: 20
- Memory: 128GB
- Hard Drive Type: SSD
- AWS: r3.4xlarge

## Additional Considerations for Pre-Conversion Services

When enabling the [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.

## PrizmDoc Application Services

The PrizmDoc Application Service (PAS) can run on a wide variety of server configurations. It will automatically scale to utilize available CPU 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.

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

- CPU Cores: 1
- Memory: 0.5 GB (Linux) / 1 GB (Windows)
- Drive Type: SSD
- AWS: t2.nano (Linux) / t2.micro (Windows)

## Suggested Requirements

- CPU 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

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

- CPU Cores: 2
- Memory: 4 GB (Linux) / 8 GB (Windows)
- Drive Type: SSD
- AWS: t2.medium (Linux) / t2.large (Windows)

## Suggested Requirements

- CPU Cores: 2
- Memory: 8 GB
- Drive Type: SSD
- AWS: m4.large

## Licensing

This section contains the following information:

- [Evaluation Licensing](#)
  - [Command-Line Mode](#)
  - [Troubleshoot Evaluation Licensing](#)
    - [No Internet Connection](#)
    - [Exceeded Installation Limit](#)
    - [License Expired](#)
    - [System is Already Licensed for Evaluation](#)
- [Deployment Licensing](#)
  - [Node-Locked](#)
    - [Without an Internet Connection](#)
  - [OEM](#)
  - [Cloud](#)
  - [Command-Line Mode](#)
- [Feature Licensing](#)

## Evaluation Licensing

This section contains the following information:

- [Command-Line Mode](#)
- [Troubleshoot Evaluation Licensing](#)

- Exceeded Installation Limit
- License Expired
- System is Already Licensed for Evaluation

## Command-Line Mode

The Prizm Licensing Utility can be used in the command-line mode for obtaining and installing evaluation licenses.

### Evaluation Licensing

#### Obtaining and Installing License from the Service

##### Usage:

```
eval get <e-mail> [requestExtension requestInstallation outputUrl]
```

##### Parameters:

- **<e-mail>** – e-mail address used to register for a trial. **Required.**
- **requestExtension** – A flag initiating a request for evaluation extension in case if evaluation license expired. *Optional.*
- **requestInstallation** – A flag initiating a request for additional installation in case of exceeding the limit of installations. *Optional.*
- **outputUrl** – A flag to output URL that can be used for licensing through the web portal in case of connectivity error. *Optional.*

##### Result codes:

- **0** – Success
- **Non-zero** – Failure

##### Examples:

- The following example demonstrates obtaining and installing an evaluation license:

```
java.exe -jar plu.jar eval get johndoe@acmecorp.com
```

- The following example demonstrates obtaining and installing an evaluation license with error handling to automatically request evaluation extension, another installation or the URL output to be used for licensing through the web portal:

```
java.exe -jar plu.jar eval get johndoe@acmecorp.com requestExtension requestInstallation outputUrl
```

#### Installing License Generated through the Web Portal

##### Usage:

```
eval write <license key>
```

##### Parameter:

- **<license key>** – License key generated through the web portal. **Required.**

- **0** – Success
- **Non-zero** – Failure

### Example:

- The following example demonstrates installing evaluation license generated through the web portal:

```
java.exe -jar plu.jar eval write 2.0.YourEvaluationLicenseKey
```

## Troubleshoot Evaluation Licensing

There are a few situations that may cause the request for an Evaluation License to fail. Use the table below to locate the appropriate troubleshooting topic based on the error message presented by the Prizm Licensing Utility:

Error Message	Resolution Topic
Application could not reach licensing services	No Internet Connection
You have exceeded the limit of evaluation installations	Exceeded Installation Limit
Your license has expired	License Expired
This system is already licensed for evaluation under another user account	System is Already Licensed for Evaluation

## No Internet Connection

Evaluation licensing can still be used on a machine without an available connection to the Internet. The process to acquire an Evaluation License in this situation does require a few additional steps, however. These steps are described below.

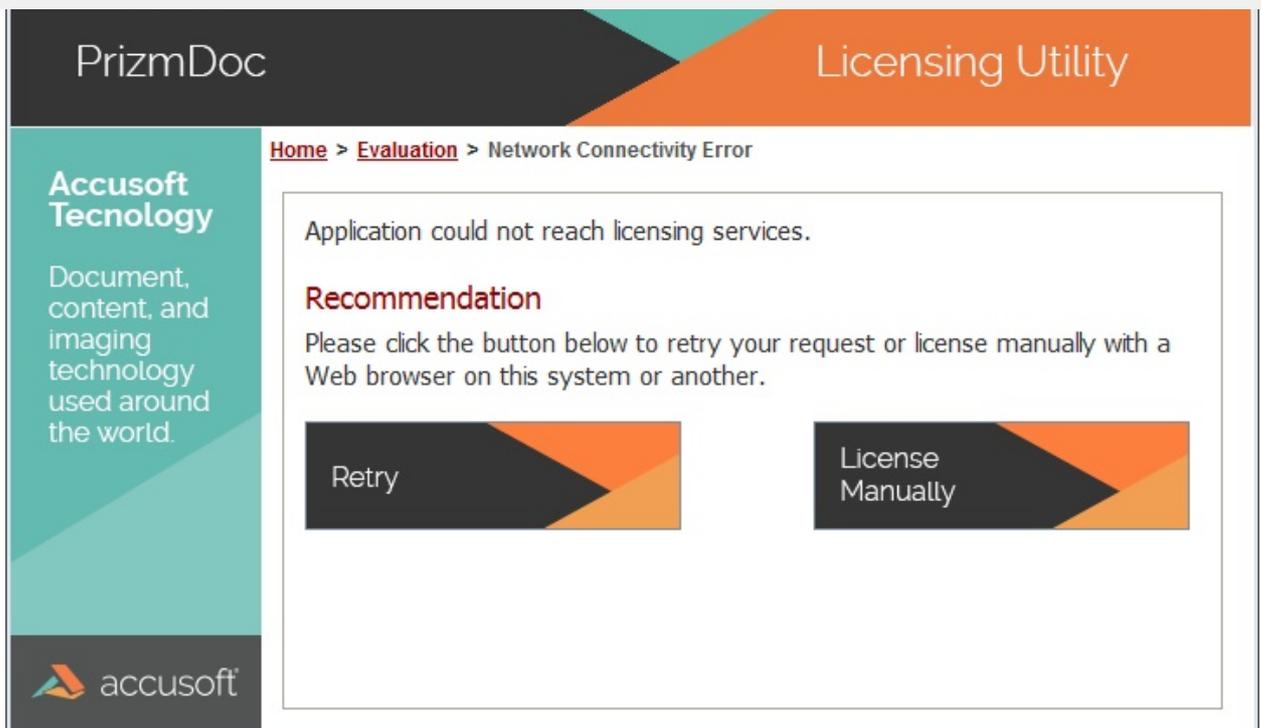
 If you have not already done so, please **complete Steps 1-5** in the Evaluation Licensing topic before proceeding with the steps below.

1. Begin the Manual Licensing Process:

The Manual Licensing Process may only be started once the Prizm Licensing Utility detects that a connection to the licensing services cannot be established. The connection is attempted when the **Request Evaluation** button is clicked on the main Evaluation screen.

The screen below is what you will see if the machine on which you are evaluating Prizm does not have an Internet connection.

Click the **Retry** button if you are aware of an Internet connection issue that has been resolved. Otherwise, click the **License Manually** button to begin the Manual Licensing Process.

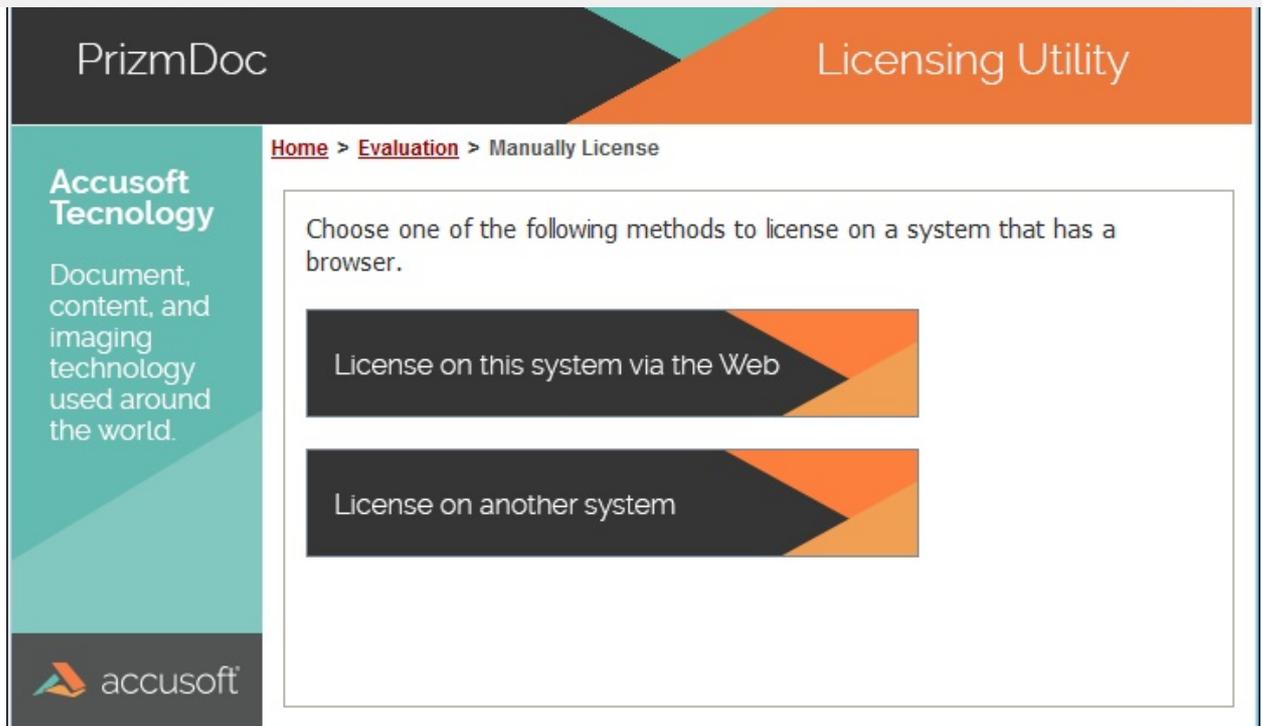


## 2. Access the Evaluation Licensing Website:

Next, you will need to go to the Evaluation Licensing website to obtain your Evaluation license. The URL to this website is provided by the Prizm Licensing Utility. It is important that the **entire** URL is used. You have two options for getting this URL:

- **License on this system via Web** - Choosing this option will open the default web browser on your machine and navigate directly to the Evaluation Licensing website. 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 Evaluation Licensing website. In Windows environments, these files can be double-clicked to open the default web browser and navigate directly to the Evaluation Licensing 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 evaluation device does not have any connection to the Internet.

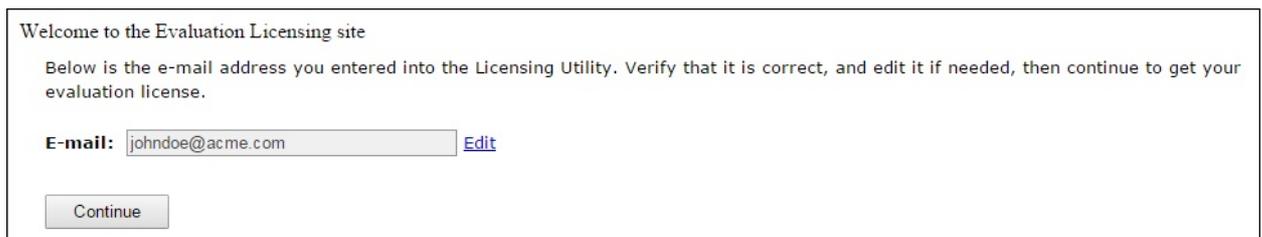
Click the button that is the best option in your situation to access the Evaluation Licensing website.



### 3. Verify Your E-mail Address and Continue:

Once at the Evaluation Licensing website, the e-mail address you entered into the Prizm Licensing Utility will already be pre-populated. If you notice an error in your e-mail address, you can correct it at this time by clicking the **Edit** link.

Click the **Continue** button to obtain your Evaluation License.



### 4. Transfer the License to the Prizm Licensing Utility:

If your Evaluation license was acquired successfully, you should see the message below. Otherwise, please see the Evaluation Licensing Troubleshooting topic for more information on these potential problems.

The string of alpha-numeric characters shown in the lower box is your Evaluation license. This information must be transferred back to the Prizm Licensing Utility running on your evaluation machine.

If you selected the option to **License on this system via Web** in **step 3**, you can simply copy the license information to your clipboard to transfer it to the Prizm Licensing Utility. If you are accessing the Evaluation Licensing website from another machine, it is recommended to download the license information to a file, and transfer the file to the evaluation machine.

Success! Your evaluation license is now available. It will expire on 3/21/2016.

To install the evaluation license, the information below must be entered into the Licensing Utility running on your device. This evaluation license will only work on the exact device from which you previously obtained the URL to this site.

**Choose an option to transfer the license:**

- [Download the License](#) to a file.
- Or, copy the text below to your clipboard and transfer it using another method.

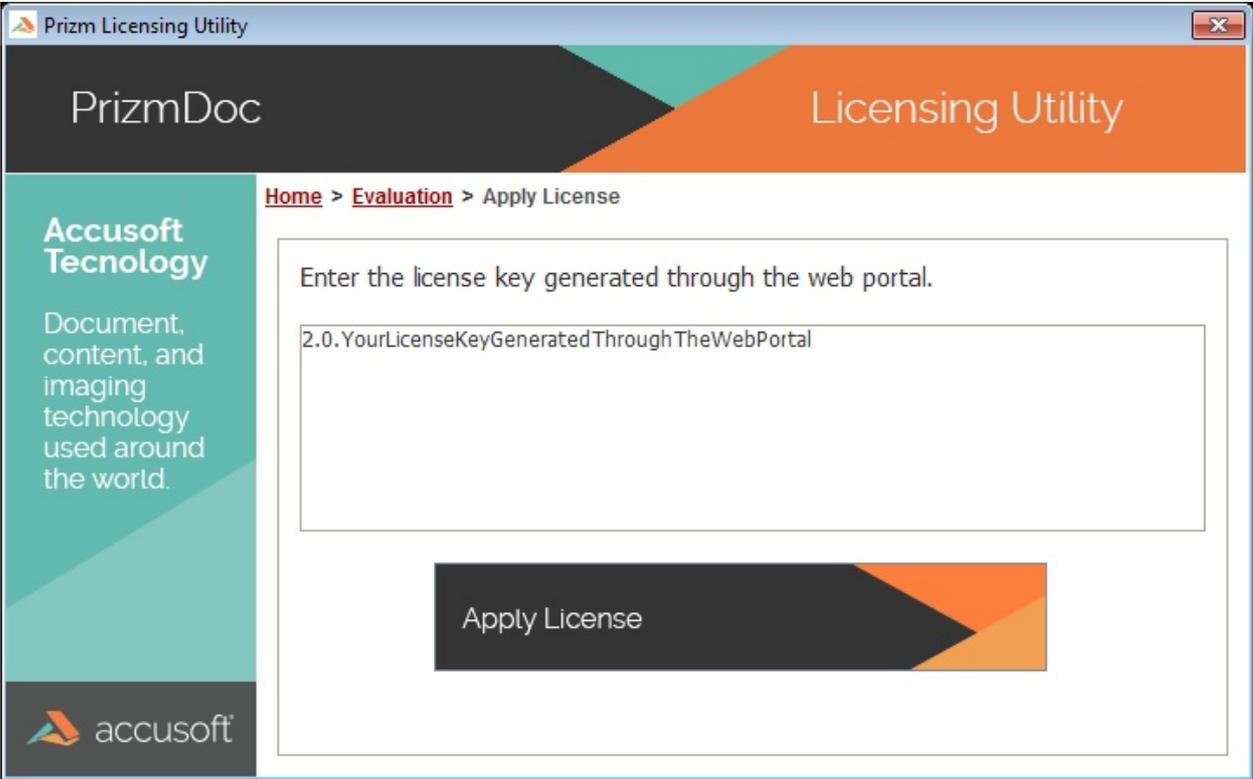
```
2.0.EWAW7C7taXsSz1bSsyQxAXjTzuRXFoMOM0zpaPke7yMufTke3s3uMg3PipMui03TFCiJA0a1FSMpMez1zJ70FS30iXbW5TRPFPiy7Ca9bFXjgASRub1iTiTF1jTRPM
```

5. Enter the License into the Prizm Licensing Utility:

Once back to the Prizm Licensing Utility running on your evaluation machine, you can paste the license information into the awaiting text area.

If the Prizm Licensing Utility was closed after you left it to go to the Evaluation Licensing site, you can restart the application and perform all the previous steps again to return to this screen in the Prizm Licensing Utility. You **do not** need to repeat the steps on the Evaluation Licensing website.

Enter the license information and click the **Apply License** to apply the Evaluation License on the current machine.



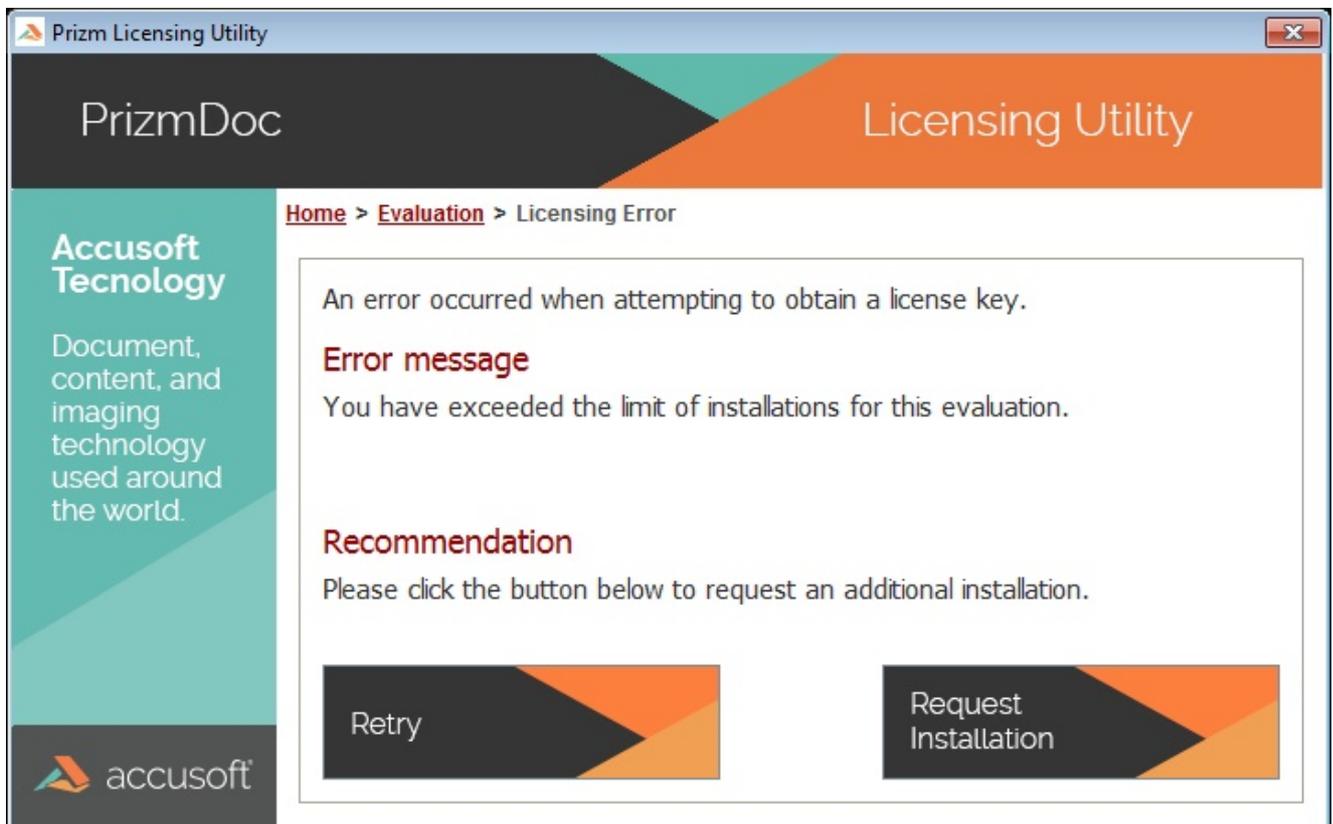
## EXCEEDED INSTALLATION LIMIT

You may see this error returned from both the Prizm Licensing Utility and Evaluation Licensing website. It means that you have obtained maximum number of Evaluation Licenses for a specific product using your Evaluation e-mail address.

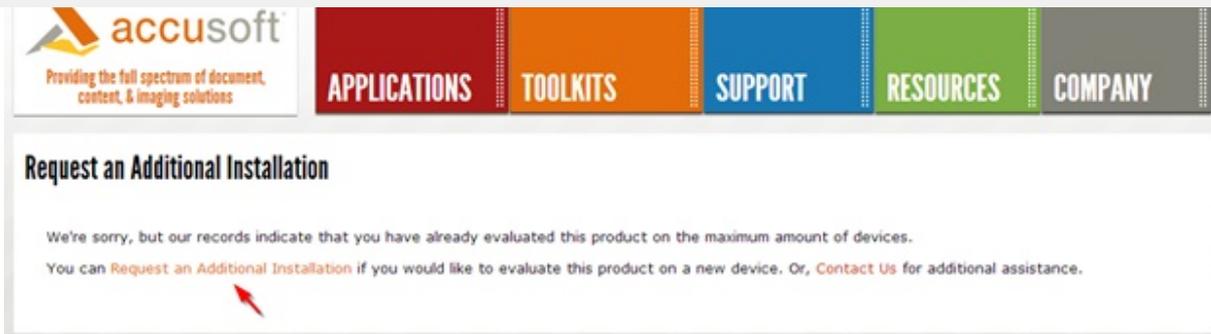
There are a couple of options for resolving this:

- **Requesting an Additional Installation** - Selecting this option will send a request to Accusoft Sales to add an additional installation for your Evaluation License. This will allow you to obtain an Evaluation License for a new machine. These requests are reviewed by Accusoft Sales staff, and are usually processed within about one business day.
- **Purchasing a Deployment License** - You may visit [www.accusoft.com](http://www.accusoft.com) for pricing information and to get in contact with Accusoft Sales about purchasing a Deployment License.

Prizm Licensing Utility: Exceeded installation limit message:



Evaluation Licensing website: Exceeded installation limit message:



## License Expired

You may see this error returned from both the Prizm Licensing Utility and Evaluation Licensing website. It means that you have previously obtained an Evaluation License for your machine, using your Evaluation e-mail address, which has since expired.

There are a couple of options for resolving this:

- **Requesting an Extension** - Selecting this option will send a request to Accusoft Sales to extend the evaluation period of your license. These requests are reviewed by Accusoft Sales staff, and are usually processed within about one business day.
- **Purchasing a Deployment License** - You may visit [www.accusoft.com](http://www.accusoft.com) for pricing information and to get in contact with Accusoft Sales about purchasing a Deployment License.

Prizm Licensing Utility: Expired license message:

The screenshot shows the PrizmDoc Licensing Utility interface. At the top, the PrizmDoc logo is on the left and 'Licensing Utility' is on the right. Below the logo is a teal sidebar with the text 'Accusoft Tecnology' and 'Document, content, and imaging technology used around the world.' The main content area has a breadcrumb trail: 'Home > Evaluation > Licensing Error'. The error message states: 'An error occurred when attempting to obtain a license key. Error message: Your license has expired. Recommendation: Please click the button below to request an evaluation extension.' There are two buttons: 'Retry' and 'Request Extension'.

Evaluation Licensing website: Expired license message:

The screenshot shows the Accusoft website navigation bar with 'MY CART' and search, currency, and social media icons. The main menu includes 'APPLICATIONS', 'TOOLKITS', 'SUPPORT', 'RESOURCES', and 'COMPANY'. Below the menu, a 'Request an Extension' section contains the text: 'We're sorry, but our records indicate that you have already received an evaluation license for this product, which has since expired. You can Request an Extension if you would like more time to evaluate this product. Or, Contact Us for additional assistance.' A red arrow points to the 'Request an Extension' link.

## System is Already Licensed for Evaluation

You may see this error returned from the Prizm Licensing Utility:

The screenshot shows the PrizmDoc Licensing Utility interface. At the top, there is a navigation bar with 'PrizmDoc' on the left and 'Licensing Utility' on the right. Below the navigation bar, there is a breadcrumb trail: 'Home > Evaluation > Licensing Error'. On the left side, there is a teal sidebar with the 'Accusoft Tecnology' logo and the text 'Document, content, and imaging technology used around the world.' The main content area contains the following text:

An error occurred when attempting to obtain a license key.

**Error message**  
This system is already licensed for evaluation under another user account.

**Recommendation**  
Please click the button below to retry your request or contact support for more information.

At the bottom of the main content area, there are two buttons: 'Retry' and 'Contact Support'. The 'Retry' button is on the left and the 'Contact Support' button is on the right. Both buttons have a dark background with a teal and orange arrow pointing to the right.

It means that you have used the wrong email address or the machine you are using has previously been licensed for evaluation with someone else's email address.

There are a couple of options for resolving this:

- **Verify the correct email address** - Verify the correct email address to use by referencing the email you were sent at the initial download time. Alternatively, you can contact Support to verify you are using the correct email address.
- **Use the email address associated with that machine's install or change machines** - If the email is correct, that means the machine you are using is already associated with another installation, so you may either use the email associated with that machine (this usually requires contacting Support), or change machines.

## Deployment Licensing

This section contains the following information:

- Node-Locked
  - Without an Internet Connection
- OEM
- Cloud
- Command-Line Mode

## 1. Obtain Licensing Information

You will receive an e-mail with your Software Registration Information. This message will contain a link to the Customer Portal, where you can obtain your licensing information for PrizmDoc. You will need to obtain a few pieces of information from the Customer Licensing Portal:

<b>Solution Name</b>	This string is used to validate the configuration file.
<b>Configuration File</b>	This file contains the information the Prizm Licensing Utilities Require in order to obtain a license for your system.
<b>Access Key</b>	For users that purchase 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 will use the next license in your pool of licenses. This will be covered in more detail below.

 Depending on the type of license you purchased, the Deployment Licensing instructions may vary. This guide is designed for Standard Licensing instructions for the Viewer and Server Distributions.

## 2. Download and Install the Product

 If you have already installed the product for evaluation or other purposes you can skip this step.

The PrizmDoc product can be downloaded from the Accusoft Web site, [PrizmDoc](#). After you have downloaded the appropriate product installer, run the installer according the versions installation instruction.

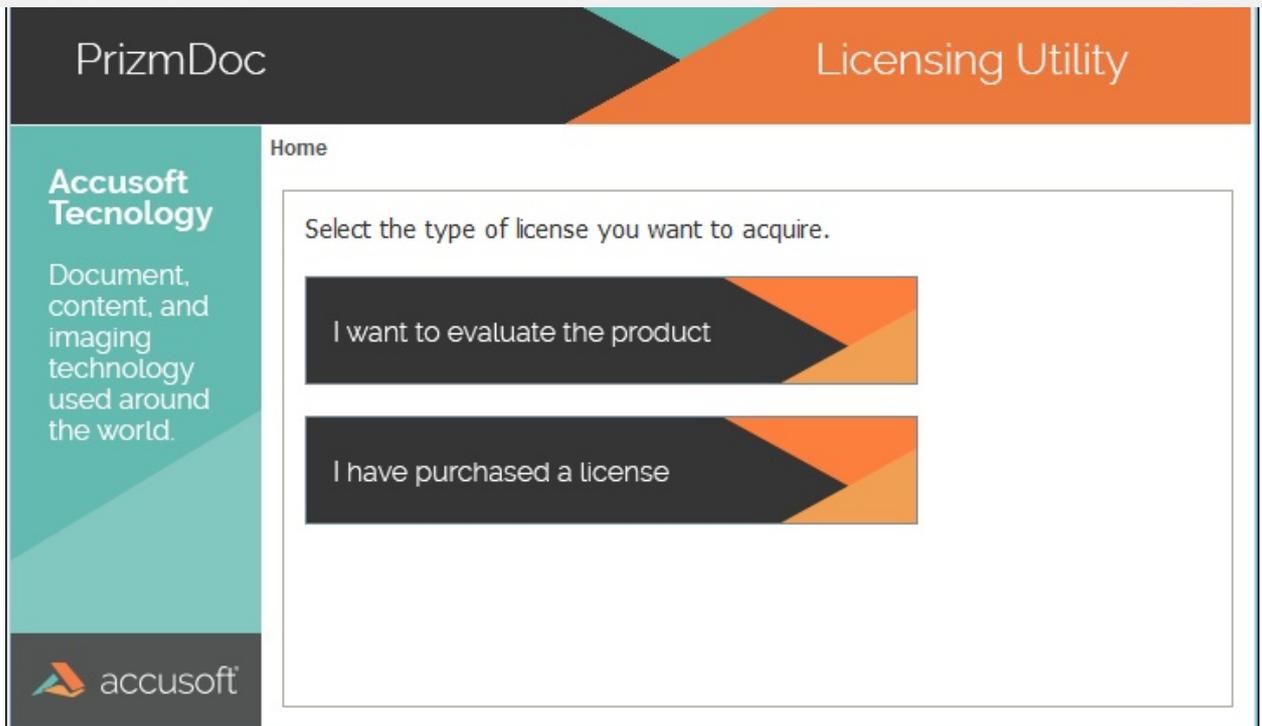
## 3. Run the Prizm Licensing Utility

The installer will run the Prizm Licensing Utility automatically as one of the final steps in the installation process. If the installer completed successfully, but you did not see the Prizm Licensing Utility, or closed it accidentally, you can launch the utility.

On Windows - The Prizm Licensing Utility can be accessed from the Start menu (**Start > Programs > Accusoft > PrizmDoc > Prizm Licensing Utility**).

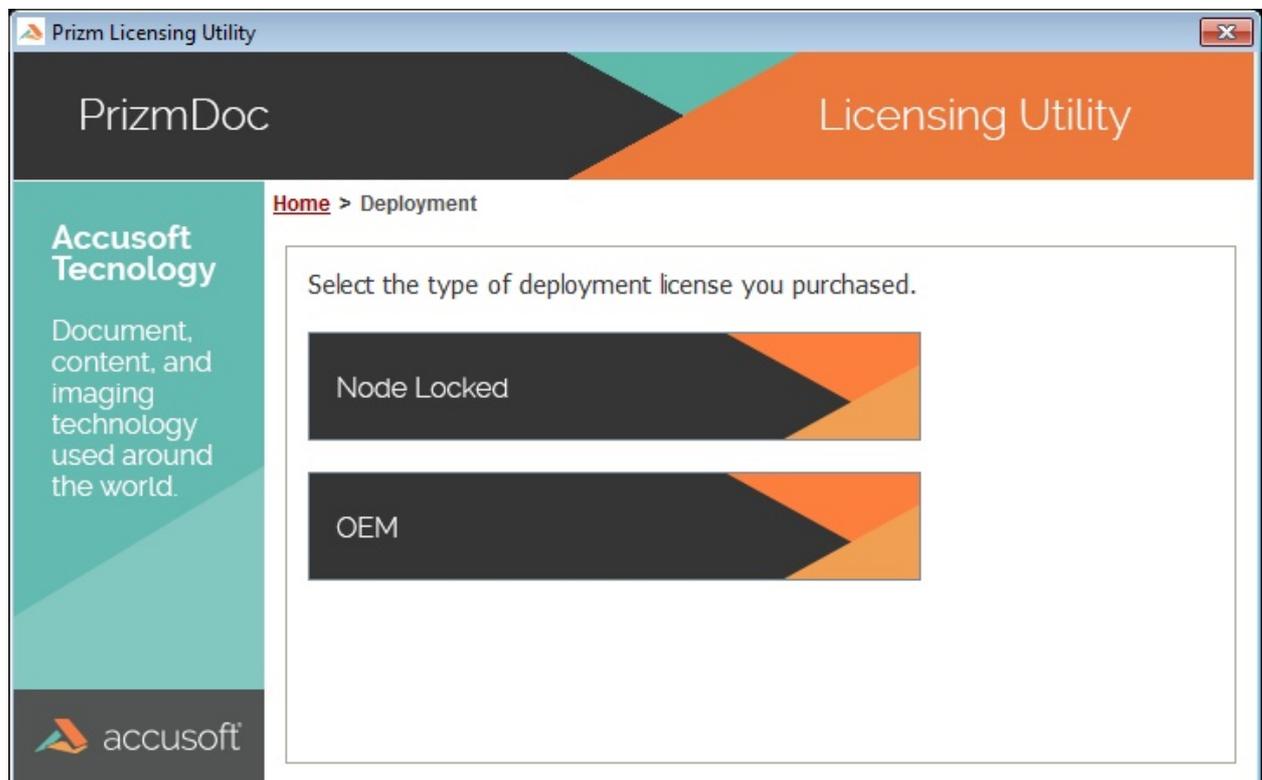
## 4. Select Deployment Option

The Prizm Licensing Utility will provide options for obtaining both Evaluation and Deployment licensing. For this walk-through, we're using Deployment licensing, so click the **I have purchased a license** button:



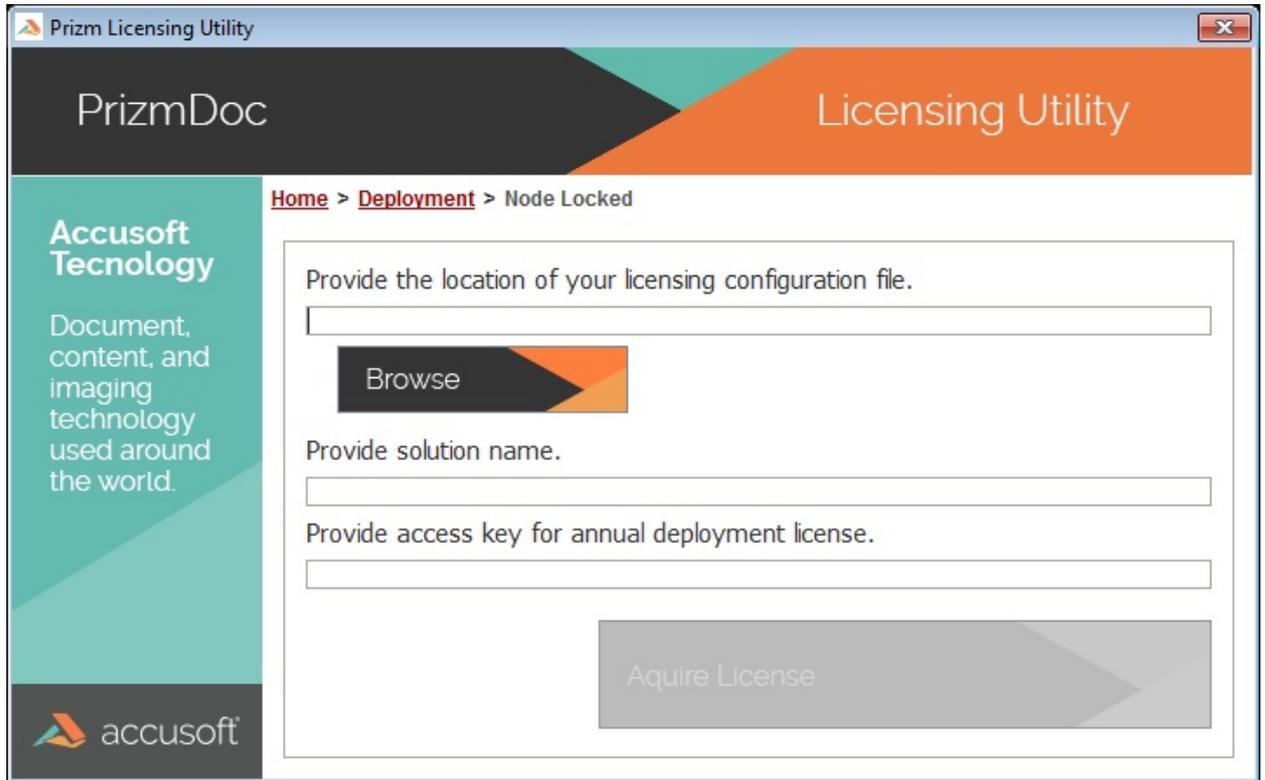
### 5. Select Node-Locked Option

The Prizm Licensing Utility is designed to assist with all licensing options. This guide is outlining how to license PrizmDoc with a purchased Server or Viewer License. Click the **Node-Locked** option:



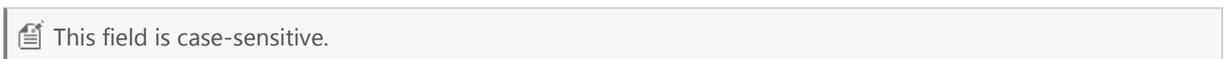
### 6. Provide Configuration File

browse to the file using the **Browse** option:



## 7. Enter Solution Name

The Solution Name was a second piece on information to be acquired from the licensing portal. The value that was assigned to your distributions should be entered into the text box defined.



PrizmDoc Licensing Utility

Home > Deployment > Node Locked

Accusoft Tecnology  
Document, content, and imaging technology used around the world.

accusoft

Provide the location of your licensing configuration file.  
C:\Downloads\YourSolutionName\_config.txt  
Browse

Provide solution name.  
Your Solution Name

Provide access key for annual deployment license.

Aquire License

### 8. (Optional) Enter your Access Key

The use of an Access Key is not required. The Access Key is a unique identifier for each distribution license purchased. If an access key is not provided, the Prizm Licensing Utility will use the next license available in your licensing pool.

 If you purchased an Annually Licensed version of PrizmDoc, it is recommend that you supply an Access Key. If you have multiple keys 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.

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.  
C:\Downloads\YourSolutionName\_config.txt  
Browse

Provide solution name.  
Your Solution Name

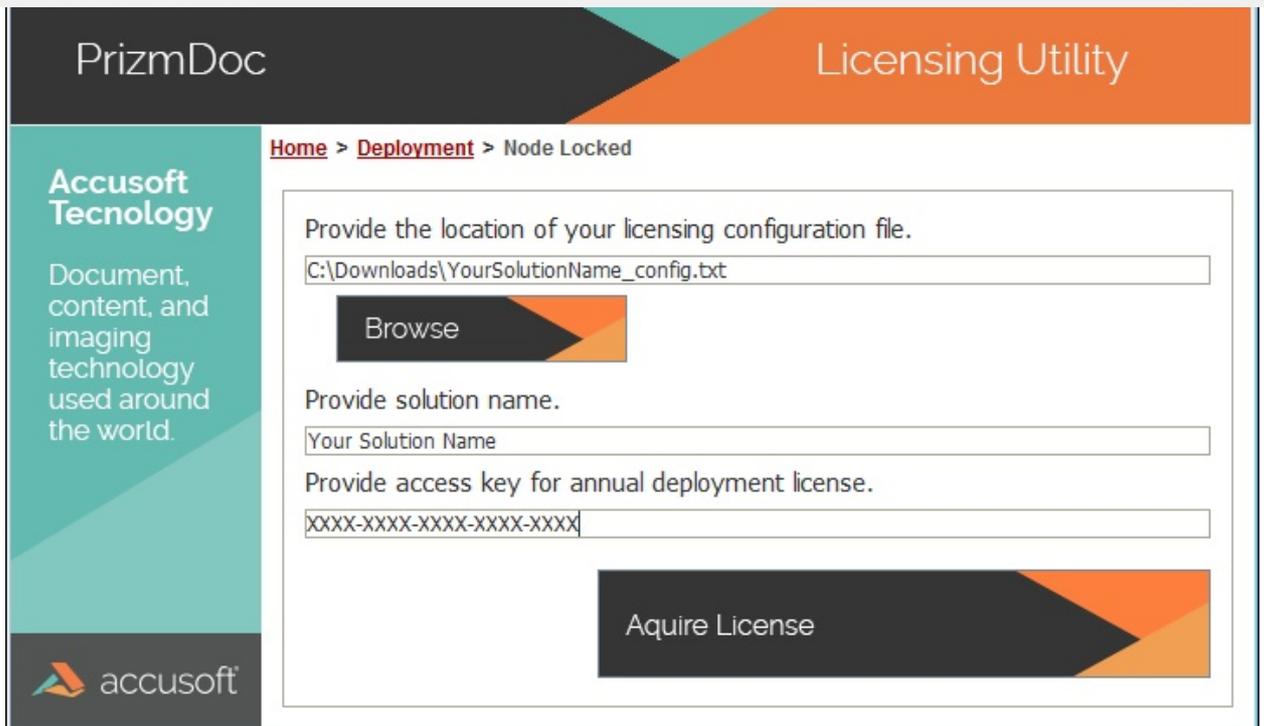
Provide access key for annual deployment license.  
XXXX-XXXX-XXXX-XXXX-XXXX

Acquire License

### 9. Acquire a License

Once all of the information has been provided you can use the **Acquire License** button to register your system with Accusoft.

 If the system does not have access to reach the Accusoft Licensing Services then the Prizm Licensing Utility will fail to register the system. You will need to follow the manual registration instructions. Please follow the steps outlined in [Licensing on a Device without an Internet Connection](#).

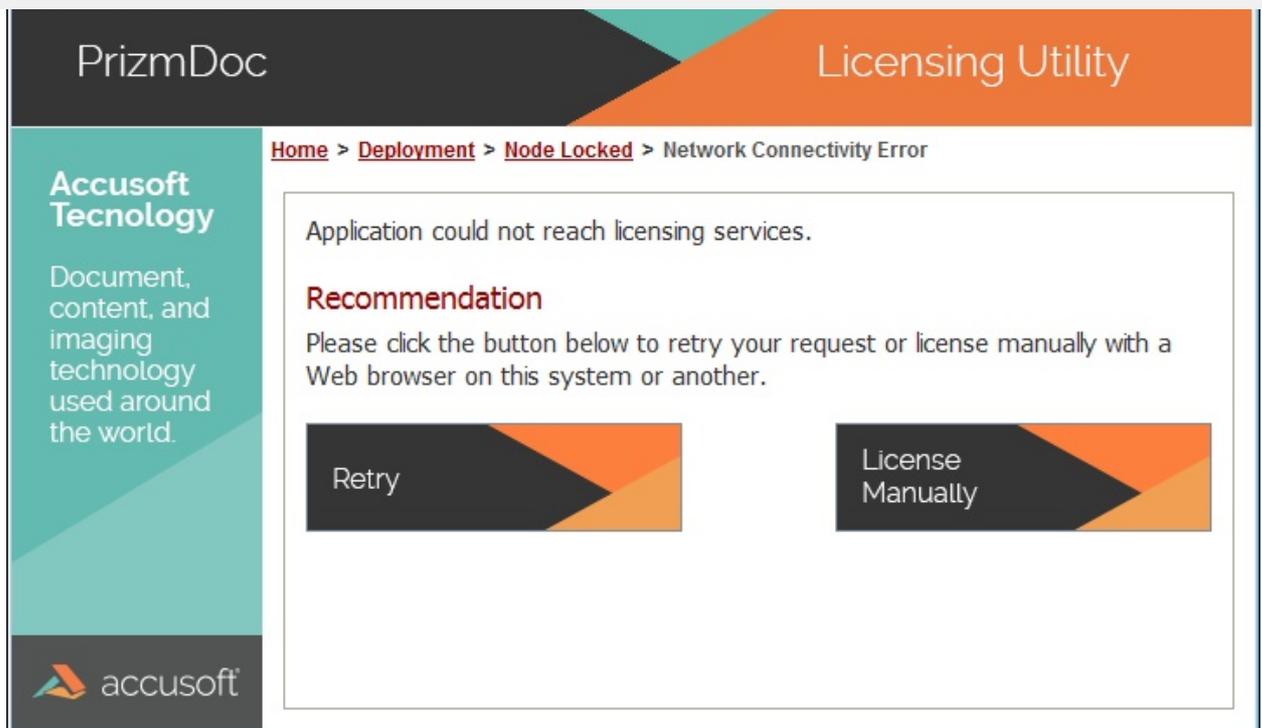


## 10. Registration Complete

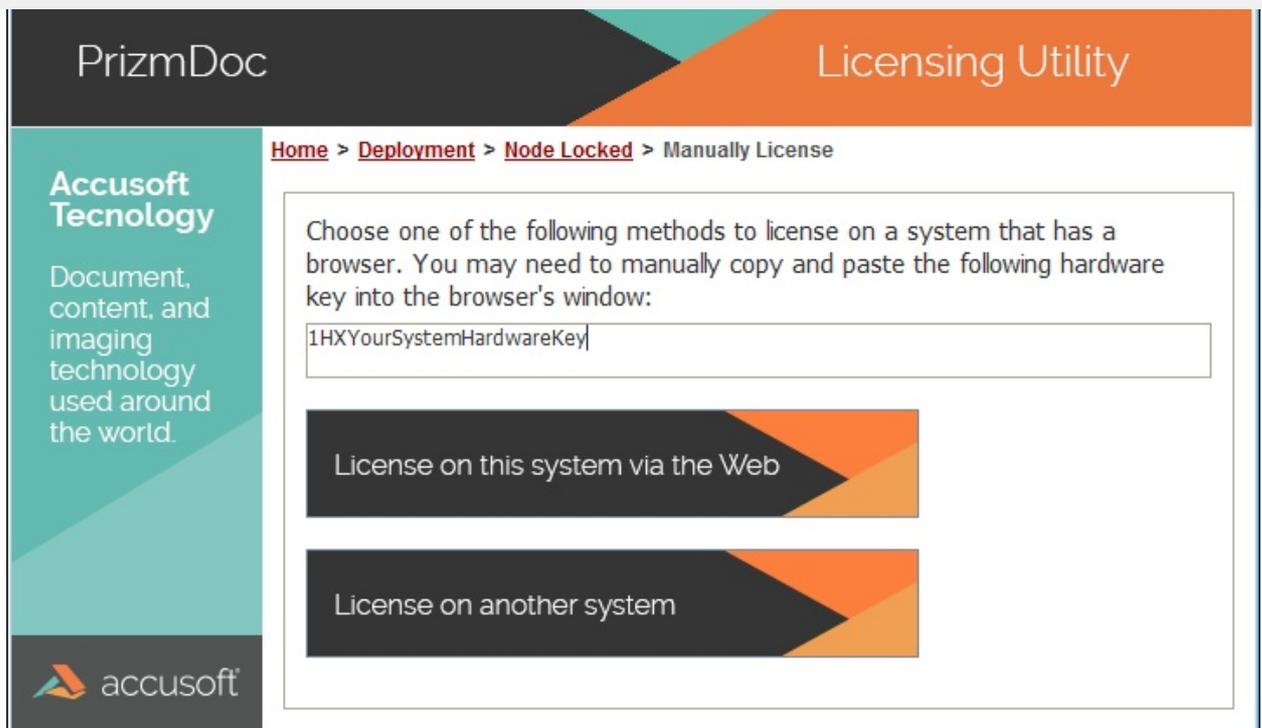
Your system has been licensed for use. If you purchased an Annual License, the Utility will display the expiration for the license which was acquired.

## Without an Internet Connection

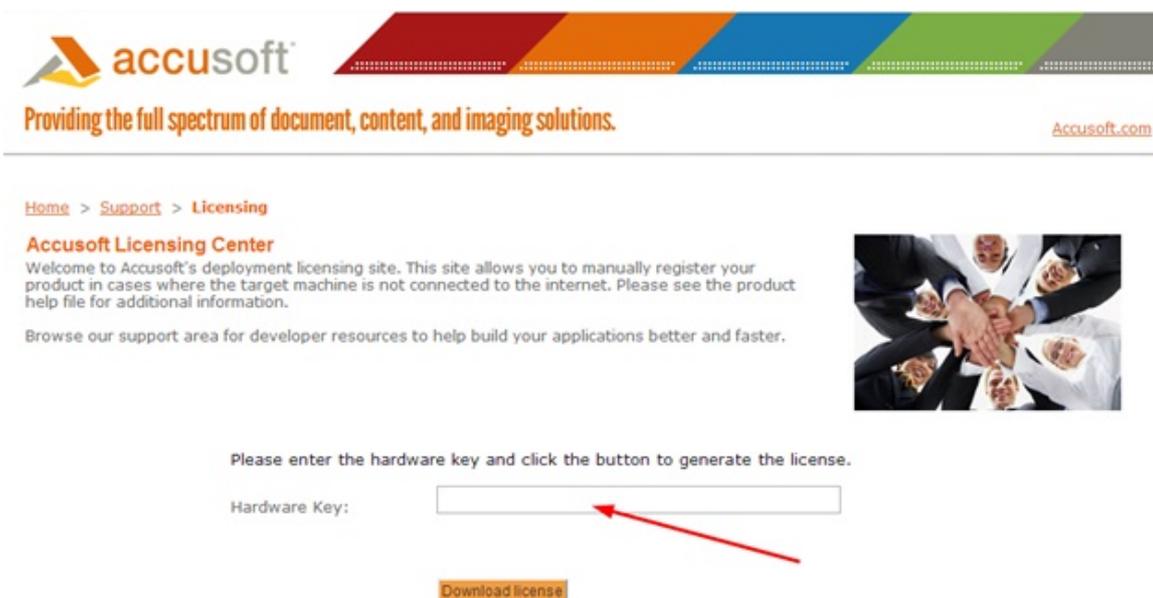
1. In the situation where the Prizm Licensing Utility is not able to contact the Licensing services, a dialog will be displayed stating that the "application could not reach the licensing services". You will have the option to retry the registration or to "License Manually". Select the **License Manually** option to proceed.



2. The Manually License dialog 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 short cut keys of Ctrl+C will copy the select data to the clipboard.
3. Next, you will need to go to the Accusoft Licensing Center to obtain your Evaluation license. The URL to this website is provided by the Prizm Licensing Utility, <https://licensing.accusoft.com/v1/WebDeployUser/WebDeployUser.aspx>. You have two options for getting this URL:
  - **License on this system via 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 enter your Hardware Key into the text box labeled **Hardware Key**.



5. Click the **Download License** button 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:

Once back to the Prizm Licensing Utility running on your system, you can paste (Use Ctrl+V to paste clipboard content into the Prizm Licensing Utility) the license information into the awaiting text area.

If the Prizm Licensing Utility was closed after you left it to go to the Accusoft Licensing Center, you can restart the

not need to repeat the steps on the Accusoft Licensing Center web site.

Enter the license information and click **Apply License** to apply the License on the current machine.

## OEM

This topic describes how to use an OEM Deployment license for PrizmDoc. OEM Deployment licenses are special Deployment licenses that must be specifically negotiated with Accusoft in order to obtain them. These licenses unlock the functionality of the product as defined in the contract and do not expire.

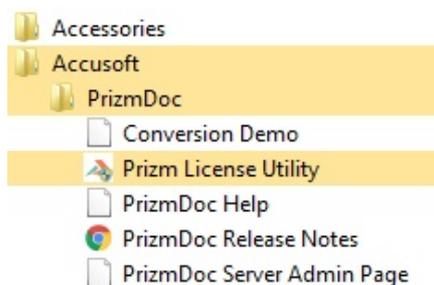
Most PrizmDoc Deployment licenses are either Annual or Perpetual.

### 1. Install **PrizmDoc**:

- If you have previously evaluated PrizmDoc and would like to use an OEM Deployment license on the same machine, there is nothing additional you need to install and you can skip to Step 2.
- If you have previously evaluated PrizmDoc and would like to use an OEM Deployment license on a different machine, you can use the same installer that was downloaded for evaluation to install the product now. Simply run the installation file to the new machine and then proceed to Step 2.
- If you have not previously evaluated PrizmDoc or you do not have access to the installer, please follow the steps in the [Evaluation Licensing](#) topic, and then proceed to Step 2 below.

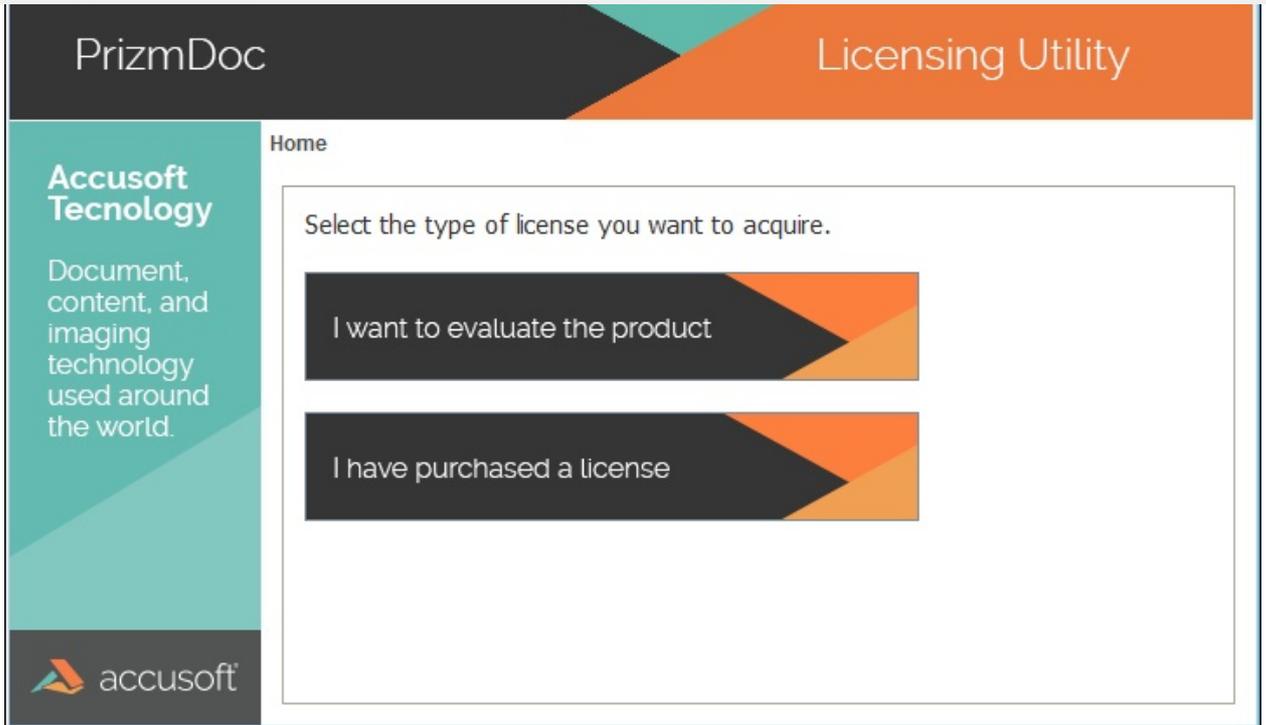
### 2. Run the **Prizm Licensing Utility**:

The installer should run the Prizm Licensing Utility automatically as one of the final steps in the installation process. If the installer completed successfully but you did not see the Prizm Licensing Utility, or if you did not need to run the installer, the Prizm Licensing Utility can be accessed from the Start Menu at the location shown below:

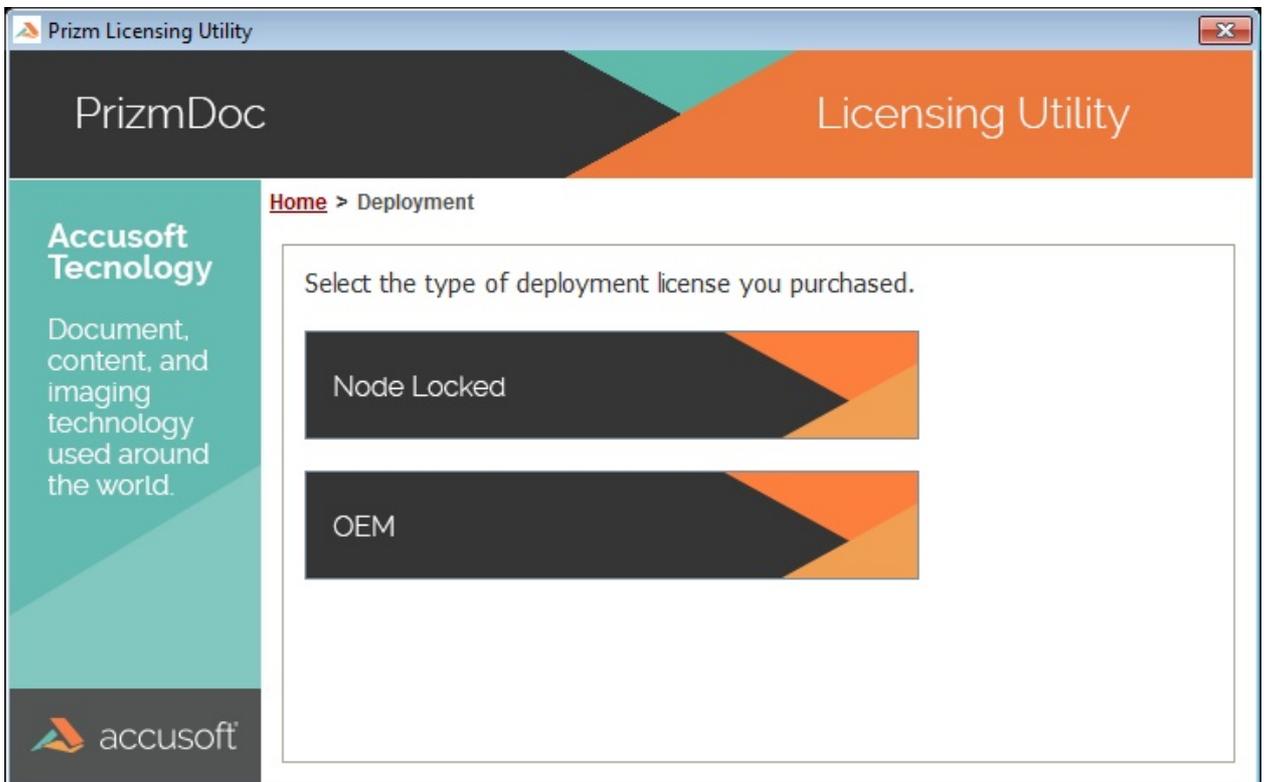


### 3. Select the Deployment Option:

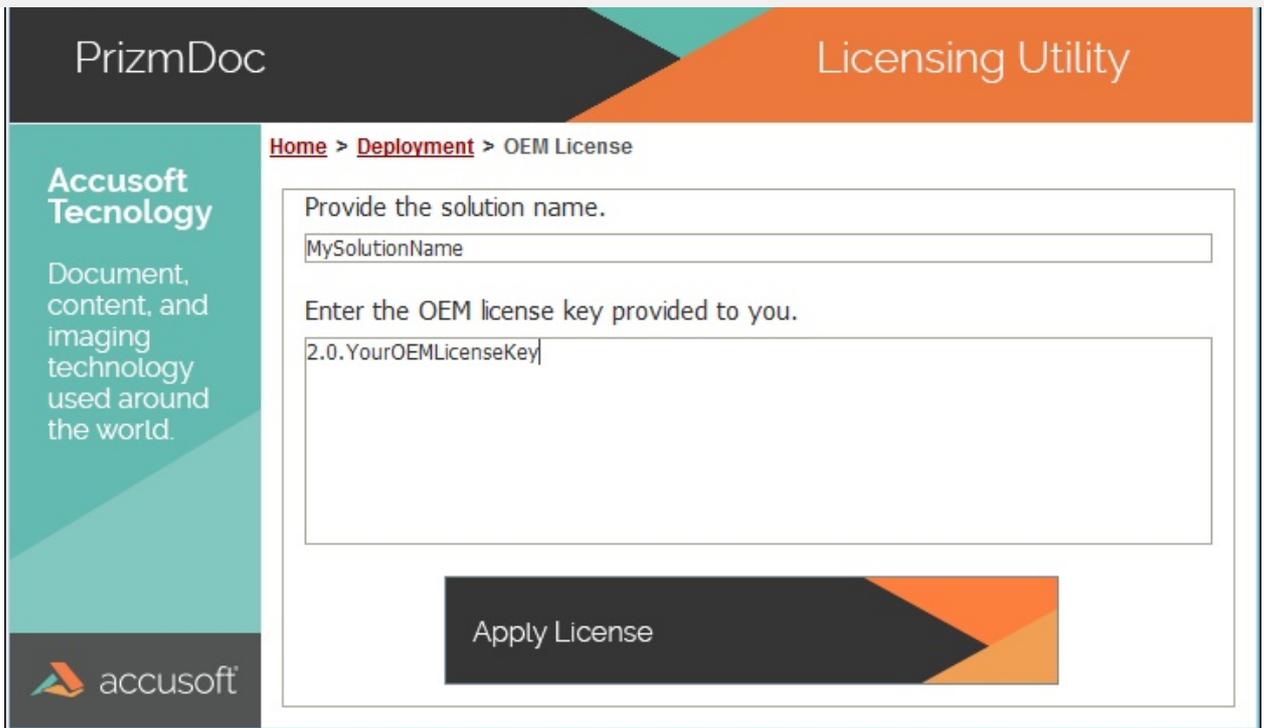
Once running, the Prizm Licensing Utility will provide options for obtaining both Evaluation and Deployment licensing. This walk-through uses OEM Deployment licensing, so click the **I have purchased a license** button:



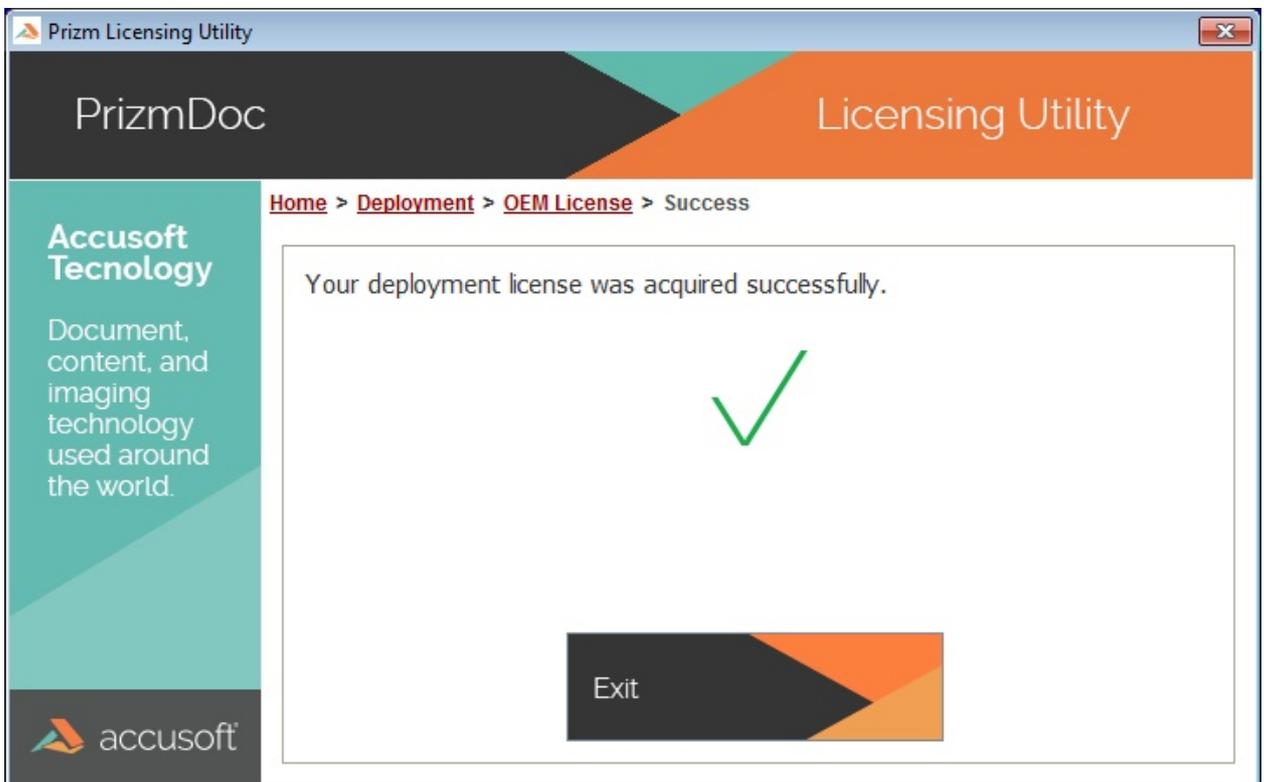
4. Click the **OEM** button:



5. Enter your **Solution Name** and **OEM License Key** values, and then click **Apply License**:



6. If your OEM Deployment license was acquired successfully, you should see the message below. Click the **Exit** button to quit the Prizm Licensing Utility and begin using your product:



## Overview

If you plan to host PrizmDoc in Amazon Web Services (AWS), you can take advantage of Cloud Licensing in order to simplify deployment and ease licensing compliance. Cloud Licensing is a lease-based licensing implementation which limits server execution based on a maximum number of licensed cores. You may purchase a single license key authorizing the use of a fixed number of cores which may be used on a single cloud server template definition. You may start up additional server instances as long as the total number of server cores is at or below the license maximum. Note that cloud licensing may be purchased as either an Annual or Perpetual license.

## Setting Up Cloud Licensing

To set up cloud licensing, you will need to do the following:

1. Set up your **Amazon S3 Account** and **S3 Bucket**.
2. Purchase **cloud-based licensing** from Accusoft.
3. Obtain **License Key** from Accusoft by providing Amazon S3 Bucket Name.
4. Load **Amazon S3** credentials.
5. Install **Accusoft License** Key using OEM Licensing instructions.

The sections below provide additional information for each step.

### 1 - Set up your Amazon S3 Account

Cloud-based licensing requires access to a cloud-based storage provider in order to register license leases. Currently, only Amazon's S3 service is supported, but other providers may be added in the future. Since you are planning to use Cloud Licensing, you are required to have an Amazon S3 account and a specific S3 Bucket which will be used as a lease repository. Please see the [Amazon S3 documentation](#) for more information on obtaining and setting up S3 Service and S3 Buckets.

### 2 - Purchase Cloud-based Licensing from Accusoft

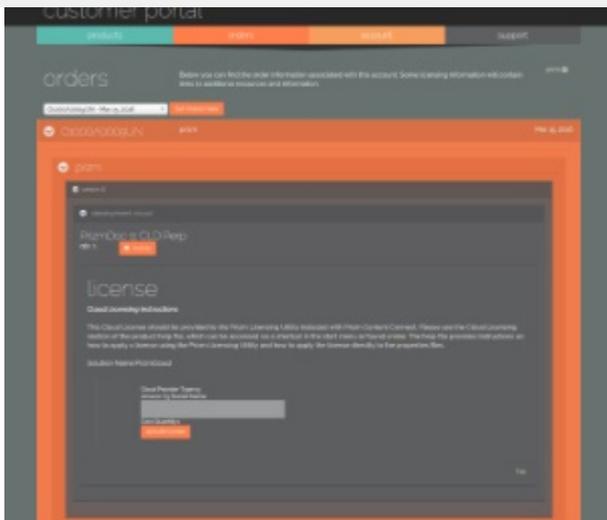
To purchase cloud-based licensing, go to Accusoft's website and fill out a [quote form online](#) or contact sales at [sales@accusoft.com](mailto:sales@accusoft.com).

At the point of purchase, you will specify the maximum number of cores to be provided by the license. You should ensure that the license will provide enough cores to enable production, support, and any ongoing development efforts.

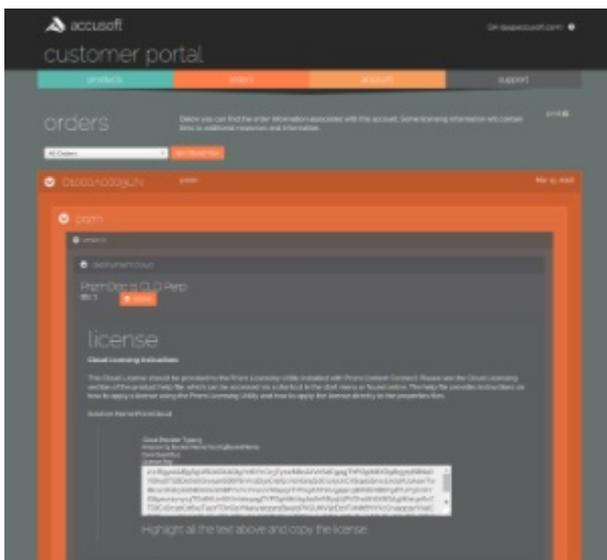
### 3 - Obtain a License Key from Accusoft

At the time of purchase, the license is not yet available for use. Prior to acquiring a license key, you will need to visit the [Accusoft Customer Portal](#) to view your purchased license and provide the S3 Bucket name that you will use for Cloud Licensing. Once the S3 Bucket name has been provided, your Cloud License key will be available for deployment. It is your responsibility to provide a valid S3 Bucket and to ensure that you have the proper credentials to access that Bucket.

- Enter your **Amazon S3 Bucket** name in the field provided and click the **Activate License** button:



- Download your **solution name** and **activated license key** to be used when configuring PrizmDoc on your server:



## 4 - Load Amazon S3 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 requires no knowledge of your S3 credentials. PrizmDoc accesses the Amazon S3 assuming that your credentials have been provided using one of the methods defined in the [Amazon SDK documentation](#) that do not require explicitly providing them to the API. As shown below, using method 1, 2, or 3 will provide access to the AWS S3.

There are several options for loading credentials. They are listed in the order of recommendation:

1. Load from **IAM roles for Amazon EC2** (if running on EC2), or
2. Load from the **shared credentials file** (~/.aws/credentials), or
3. Load from **environment variables**.

environment for the user under which PRIZMDOC will execute.

## In Linux

### Example

```
export AWS_ACCESS_KEY_ID=XXXXXXXXXX
export AWS_SECRET_ACCESS_KEY=XXXXXXXXXX
```

## In Windows

### Example

```
set AWS_ACCESS_KEY_ID=XXXXXXXXXX
set AWS_SECRET_ACCESS_KEY=XXXXXXXXXX
```

## 5 - OEM Licensing

After configuring the credentials for Amazon S3, install the license key using the instructions provided in the [OEM Licensing](#) topic.

# Command-Line Mode

The Prizm Licensing Utility can be used in the command line mode for obtaining and installing deployment licenses.

## Deployment Licensing

### Obtaining and Installing License from the Service

#### 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. *Optional.*
- **outputUrl** – A flag to output URL that can be used for licensing through the web portal in case of connectivity error. *Optional.*

#### Result codes:

- **0** – Success
- **Non-zero** – Failure

#### Examples:

- The following example demonstrates obtaining and installing 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 deployment license for the provided

```
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 deployment license with error handling to automatically output URL to be used for licensing through the web portal:

```
java.exe -jar plu.jar deploy get "C:\Path to\YourSolutionName_Config.txt" "Your Solution Name"  
outputUrl
```

## Installing License 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.**

### Result codes:

- **0** – Success
- **Non-zero** – Failure

### Examples:

- The following example demonstrates installing deployment license generated through the web portal:

```
java.exe -jar plu.jar deploy write "Your Solution Name" 2.0.YourDeploymentLicenseKey
```

## Installing OEM License

### Usage:

```
deploy write <solution name> <license key>
```

### Parameters:

- **<solution name>** – Solution name for OEM licensing. **Required.**
- **<license key>** – OEM license key. **Required.**

### Result codes:

- **0** – Success
- **Non-zero** – Failure

### Examples:

- The following example demonstrates installing OEM license:

```
java.exe -jar plu.jar deploy write "Your OEM Solution Name" 2.0.YourOemLicenseKey
```

## Feature Licensing

### Overview of PrizmDoc Feature Licensing

so that you can decide which features are important for your customers. Please note that this section applies to PrizmDoc Self-Hosted installations only. PrizmDoc Accusoft Cloud-Hosted already includes all of these features.

The following features can be licensed separately:

- **Microsoft Office Conversion** - Microsoft Office Conversion provides native rendering of Word, Excel, and PowerPoint documents for those needing these Office documents to render exactly as they would in Microsoft Office. For configuration details, refer to the following topics:
  - [Working with PrizmDoc > Developer Guide > PrizmDoc Server > How To > Natively Render Microsoft Office Documents](#)
- **Form Field Detector** - PrizmDoc automatically detects forms in both PDF and Raster documents.
  - How to use the E-Signature module with the [Form Field Detector](#).
  - Refer to the following APIs for the Form Field Detector:
    - [PrizmDoc Application Services PAS RESTful API > Developer Reference > Form Extractors](#)
    - [PrizmDoc Server RESTful API > Developer Reference > Form Extractors](#)
    - [PrizmDoc E-Signature Viewer API > Module: form-extraction](#)

## Licensing

If you are an existing customer:

- You will need to upgrade to v12.0 in order to take advantage of the new Microsoft Office Conversion and Form Field Detector features. Note that in order to use the Microsoft Office Conversion feature, you must have a Microsoft Office product (Microsoft Office 2013 or 2016 Standard Edition) installed and licensed on your server prior to installing PrizmDoc. Contact [Sales](#) to purchase a license key that includes the features that you want. Refer to the following section for details on [Getting Started with PrizmDoc](#).
- If you already have v12.0 installed, but want to enable additional features, contact [Sales](#) to purchase a license key that includes the features that you want. Once you have the right license key, repeat the licensing step you did during initial installation, supplying the new license key, and restart the PrizmDoc service. That way you don't need to reinstall the product. For information about the steps, refer to the appropriate help sections: [Getting Started with PrizmDoc](#), [Deployment Licensing](#), [Starting & Stopping PrizmDoc Server](#).

If you are a new customer:

- Contact [Sales](#) to purchase a license key that includes the features that you want. Note that in order to use the Microsoft Office Conversion feature, you must have a Microsoft Office product (Microsoft Office 2013 or 2016 Standard Edition) installed and licensed on your server prior to installing PrizmDoc. Refer to the following section for details on [Getting Started with PrizmDoc](#).

## System Configuration

This section contains the following information:

- [Overview](#)

- Configuration Options
  - Initialization Parameters
  - uiElements
  - Pre-Loaded Search Parameters
  - Configuring Skinny Comments Panel
  - Defining the View Mode
  - Digital Rights Management Configuration
  - Enabling Content Encryption
  - How to use Predefined Search
  - Localizing the Viewer
  - Using a Custom Resource Path
  - Adding Custom Image Stamps
- Configure PrizmDoc Application Services (PAS)
  - PrizmDoc Application Services (PAS) Configuration
  - PrizmDoc Application Services Database Administration & Maintenance
  - How & when to use CORS with PrizmDoc
- Configure the PrizmDoc Server
  - Central Configuration
  - Upgrade to Central Configuration
  - Configuring Ports
  - Format Detection Configuration & Use
  - Adjust Caching Parameters for PrizmDoc Server
  - Adjust Office Conversion Settings for Optimal Performance
  - Adjust Vector Conversion Settings for Optimal Performance
  - Change Encryption Keys for Public use Token Generation
  - Configure Image Frame Rendering in the PDF Conversion Service
  - Configure Log File Locations
  - Configure Microsoft Office Conversion Connectivity
  - Customize Excel Pagination Settings for PrizmDoc Server
  - Customize Excel Document View Settings for PrizmDoc Server
  - Customize Text File Encoding for PrizmDoc Server
  - Disable Excel Pagination for PrizmDoc Server
  - Implement PrizmDoc Server Caching Strategies
  - Substitute Fonts for Office Rendering Fidelity

## Overview

PrizmDoc and its components can be configured in numerous ways to address different use cases. This article will give a brief overview of some of the more common configuration cases as well as serve as a roadmap to find more information about how to configure the system to best address your needs.

- [Configuring PrizmDoc Application Services](#)
- [Configuring PrizmDoc Server](#)

## Configuring PrizmDoc Samples & Viewer Applications

The Viewer is a highly customizable component that we've embedded into our web-tier applications that can be used out of the box or customized and integrated into other applications. For an overview of the Viewer's architecture, please refer to the [Viewer Overview](#).

The Viewer has several options that change the way the viewer plugin is initialized. These plugin options are formatted as a JavaScript object that is passed in when the viewer plugin is created by the client side application. These options are specific to a single instance of the viewing application, and they can be configured independently of other instances of the viewer. For a full description of the options available, refer to our [Namespace: fn API](#) topic.

Additionally, refer to our [Configuration Options](#) topic for some examples on how to configure the Viewer to suit your needs. For example, if you want to configure protection for the content in the viewer, refer to the [DRM](#) and [Content Encryption](#) articles.

## Configuring PrizmDoc Application Services

PrizmDoc Application Services, or PAS, is a service that provides a layer of application-level logic for the PrizmDoc Viewer. For an overview of how PAS integrates into most systems please refer to the [PAS Overview](#).

PrizmDoc Application Services configuration should begin with taking a look at the [PAS Configuration article](#), which will address most of the common configuration questions.

If you're new to PAS, but are an existing PrizmDoc customer, take a look at our [migration](#) topic. There you can find more information about how to [override specific PAS routes](#) if, for example, your application was using custom storage for your markup files. If your application used much of the logic from our earlier web-tier samples, refer to the [Legacy Mode section in our PAS Configuration](#) topic to enable support for markup files created with those samples.

We also have examples about how to [configure PAS in your server's entry point](#); however, if the proxying examples are not feasible for your deployment, you can decide if you need to use [CORS with PAS](#).

If your application will require more bandwidth than a single server can provide, review our comments regarding [running PAS on multiple servers](#).

## Configuring PrizmDoc Server

PrizmDoc Server is a collection of different services that can accomplish, and be used by, various workflows. In past iterations, there were multiple configuration files that an administrator would work with to configure their system. PrizmDoc Server now relies on a single [Central Configuration](#) for both Windows and Linux installations.

While legacy configuration files are still compatible, existing PrizmDoc customers will want to [upgrade their legacy configuration to the new central configuration](#) for greater simplicity. Please refer to the table in that topic to identify values that may have been set in your deployment and their corresponding values in Central Config.

Look to our [Security Guidance](#) if you want more information on how to take steps to secure the [Viewing](#)

network infrastructure information, see our topic on [Changing Encryption Keys](#) to non-default values.

By default, a single server with an instance of the PrizmDoc Server RESTful API needs only one port that is open to the rest of your network. The number of ports required increases to two with a multi-server configuration. However, PrizmDoc server consists of multiple micro-services that require a range of open, local ports to function. Refer to [Configuring Ports for the PrizmDoc Server](#) for more information and how to configure them.

PrizmDoc Server works most efficiently when it can rely on its cached content to service viewing, burning, and conversion requests. If the Server has already cached a document, the time between the initial request and the first page loads in the viewer is greatly reduced; however, if your usage means that the majority of viewed documents will be unique or you're not able to cache them, there may be very little to no benefit to maintaining a Server cache. Please see our [Caching Strategies](#) for a more detailed look at how and why PrizmDoc Server generates a cache and how that process can be optimized for your application. Additionally, see our [Caching Parameters](#) topic for greater detail on how to configure the duration of PrizmDoc Server's cached content, where it is stored, and whether or not it is used at all. Please note that PrizmDoc Server's caching mechanism is not intended for long-term storage of converted content.

If the majority of content intended for viewing and conversion is Microsoft Office content (Word, Excel, Powerpoint), you may want to adjust some of the parameters related to Office conversion. Our [Office Conversion Settings](#) topic goes into some detail about which settings can be adjusted as well as the pros and cons of adjusting those parameters. Additionally, PrizmDoc Server uses some default values when converting Excel content for viewing. Depending on the layout of your content, those values may need to be adjusted or disabled entirely. See our [Excel Pagination](#) topic for more information.

If there are additional configuration topics that you have questions about, or you think are missing from the above, please contact [support@accusoft.com](mailto:support@accusoft.com) or visit <https://www.accusoft.com/support/>.

## Configure the Viewer

This section contains the following information:

- [Configuration Options](#)
  - [Initialization Parameters](#)
  - [uiElements](#)
  - [Pre-Loaded Search Parameters](#)
  - [Configuring Skinny Comments Panel](#)
  - [Defining the View Mode](#)
  - [Digital Rights Management Configuration](#)
  - [Enabling Content Encryption](#)
  - [How to use Predefined Search](#)
  - [Localizing the Viewer](#)
  - [Using a Custom Resource Path](#)
  - [Adding Custom Image Stamps](#)

## Configuration Options

This section contains the following information:

- [Initialization Parameters](#)
- [uiElements](#)
- [Pre-Loaded Search Parameters](#)
- [Configuring Skinny Comments Panel](#)
- [Defining the View Mode](#)
- [Digital Rights Management Configuration](#)
- [Enabling Content Encryption](#)
- [How to use Predefined Search](#)
- [Localizing the Viewer](#)
- [Using a Custom Resource Path](#)
- [Adding Custom Image Stamps](#)

## Initialization Parameters

The following table contains a list of some of the configurable options when initializing the Viewer plugin. For additional configuration options, refer to the [Namespace: fn](#) API topic.

Parameter	Data Type	Description
annotationID	String	<p>Specifies the annotation file to be used within the Viewer.</p> <p>For example:</p> <ol style="list-style-type: none"><li>1. When the annotation is saved, the Viewer prompts the user to provide the annotation name (example: myTestAnnotation).</li><li>2. The annotation file is saved with this naming convention: &lt;documentID&gt;_&lt;annotationName&gt;.xml (example: F7-92-5C-CE-28-BE-AD-6F-79-C3-FE-FF-87-FE-FA-B6-73-7F-F5-92_0_myTestAnnotation.xml).</li><li>3. When specifying the value for the annotationID, it will be the annotation name that was provided at the time of saving (example: stampAnnotation), <b>not</b> the annotation file name (F7-92-5C-CE-28-BE-AD-6F-79-C3-FE-FF-87-FE-FA-B6-73-7F-F5-92_0_myTestAnnotation.xml).</li></ol>
autoLoadAnnotation	Boolean	If set to True, the specified annotation file will be loaded when the Viewer launches resourcePath.
debug	Boolean	If true, Viewer logs to the console. If false or no value, Viewer will not log to the console.

encryption	Boolean	Specifies whether content encryption is turned on or off.
imageHandlerUrl	String	Specifies the location of the image handler.
language	Object	An object containing the contents of the language.json file.
predefinedSearch	Object	An object containing all predefined search options. These are defined in the predefinedSearch.json file. See <a href="#">Pre-loaded Search</a> for more information.
resourcePath	String	The location of the images used in the Viewer.
serviceResponseTimeout	Integer	Indicates the response timeout interval, in milliseconds, for all services to the server. Default is 60000.
signatureCategories	String	Specifies the categories of E-Signature to add to the UI. This is a comma separated string containing the types of signatures that the application expects. When no value is passed in, the UI to select categories will be disabled. Example: "Full Signature,Initials,Name,Title"
template	Object	An object containing the different templates used in the Viewer.
uiElements	Object	An object containing parameters that toggle UI elements in the Viewer. See <a href="#">uiElements</a> for more information.

## uiElements

The following is a list of configurable UI elements when initializing the Viewer plugin:

Parameter	Data Type	Description
annotateTab	Boolean	Hide or show the Annotate Tab. Default is true.
copyPaste	Boolean	Hide or show the Text Select Tool. Default is true.
download	Boolean	Hide or show the Download Button. Default is true.
esignTab	Boolean	Hide or show the E-Signature Tab. Default is true.
fullScreenOnInit	Boolean	Specifies whether the Viewer will fill the browser window when initialized. If this is not defined or set to false the Viewer will use the width and height set in viewer.css.
printing	Boolean	Hide or show the Print Button. Default is true.
redactTab	Boolean	Hide or show the Redact Tab. Default is true.
searchTab	Boolean	Hide or show the Search Tab. Default is true.
viewTab	Boolean	Hide or show the View Tab. Default is true.
advancedSearch	Boolean	Hide or show advanced search options. Default is false.

## Pre-Loaded Search Parameters

### preDefinedSearch Parameters

This object contains all predefined search options:

Parameter	Data Type	Description
highlightColor	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"
searchOnInit	Boolean	Run search on launch.
globalOptions	Object	Set the default search options for each of the predefined search terms. This is overridden by the term-level "options" parameter.  Example:  <pre>predefinedSearch : {   globalOptions: {     matchCase: false,     endsWith: false,     beginsWith: false,     matchWholeWord: false   } }</pre>
terms	Array	An array of objects that represent the search terms that will be available in the predefined search menu.  Example:  <pre>predefinedSearch : {   terms: [     {       searchTerm: "llama"     }   ] }</pre>

### predefinedSearch terms

Parameter	Data Type	Description
searchTerm	String	The search string for the term object. This is overridden by the userDefinedRegex parameter.
userDefinedRegex	String	<p>A regular expression that will be searched in place of searchTerm. 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>Example:</p> <pre>predefinedSearch : {   terms: [     {       searchTerm: "4 digits"       userdefinedRegex: "(\\d{4})"     }   ] }</pre>
selected	Boolean	Whether or not this term will be selected in the menu.
options	Object	<p>Set the search options for this term. If a parameter is not defined it will inherit the globalOptions-level parameter.</p> <p>Example:</p> <pre>predefinedSearch : {   terms: [     {       searchTerm: "Lla"       options: {         matchCase: true,         endsWith: false,         beginsWith: true,         matchWholeWord: false       }     }   ] }</pre>

```
}
```

## Configuring Skinny Comments Panel

To better accommodate the Comments view in both small and large viewers, the Viewer automatically switches between displaying the comments in their entirety to displaying only a Comments icon. When you click the Comments icon, the full comment is expanded. By default, the mode automatically switches between the two pages on the space available for viewing the document.

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 – note, this is the default option, so it does not need to be explicitly defined:

### Example

```
var pluginOptions = {
    documentID: viewingSessionId,
    language: languageItems,
    template: htmlTemplates,
    commentsPanelMode: "auto"
};
$(document).ready(function () {
    var viewerControl = $("#viewer1").pccViewer(pluginOptions).viewerControl;
});
```

## Defining the View Mode

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
```

## Digital Rights Management Configuration

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:
  - a. Change the plugin options for DRM.
  - b. 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](#)
- [Copying Text](#)
- [Printing](#)

### Document Download

**download**).

- a. 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"**).
  - a. 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**).
  - a. This can be controlled by the server-side code that generates the page.
  - b. 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 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.

## Enabling Content Encryption

This topic contains the following information:

- [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](#)
- [Disabling Content Encryption in the PrizmDoc Server via the Central Configuration File](#)
- [Disabling 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 to discern 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

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.

 This feature is not supported in IE8.

## 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.

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 options are both documented below.

 For the security conscious, toggling content encryption per viewing session is not permitted in the out of box product configuration. It must be explicitly allowed via the ServiceHost pcc.config 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

```
# 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
```

```
# 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 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

```
# 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

## 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

```
function buildViewerOptions() {
    ...
    var optionsOverride = args.pop(); // always last arg
    var options = {
        ...
        encryption: true
    };

    var combinedOptions = _.extend(optionsOverride, options);

    embedViewer(combinedOptions);
}
```

 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](#)

## Disabling Content Encryption in the PrizmDoc Server via the Central Configuration File

To disable Content Encryption in the PrizmDoc Server, 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 **false**.

```
# 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: false
```

3. Save the **changes** to the file.
4. Restart the **PrizmDoc Server** for the changes to take effect.

## Disabling Content Encryption in the Viewer

To disable encryption in the Viewer, use the Viewer's default behavior without providing the encryption option. By default, the Viewer sets the encryption value to 'false'. Should you wish not to use the encryption in the viewer options parameter, set the encryption option to false as shown below:

### Example

```
function buildViewerOptions() {
    ...
    var optionsOverride = args.pop(); // always last arg
    var options = {
        ...
        encryption: false
    };

    var combinedOptions = _.extend(optionsOverride, options);

    embedViewer(combinedOptions);
}
```

 Enabling/disabling the encryption will not work without appropriately setting the PrizmDoc Server configuration.

## How to use Predefined Search

Predefined search 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 = {
```

```

language: languageJSON,
template: htmlTemplates,
predefinedSearch: {
  highlightColor: "#ee3a8c",
  searchOnInit: false,
  globalOptions: {
    matchCase: false,
    endsWith: false,
    beginsWith: false,
    matchWholeWord: false
  },
  terms: [{
    searchTerm: "llama",
    selected: true,
    options: {
      matchWholeWord: true
    }
  },
  {
    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). PredefinedSearch.json provides several sample search terms and custom regular expressions. PredefinedSearch.json file is parsed by the web-tier and loaded in the Viewer. The following example shows you how:

### Example

```

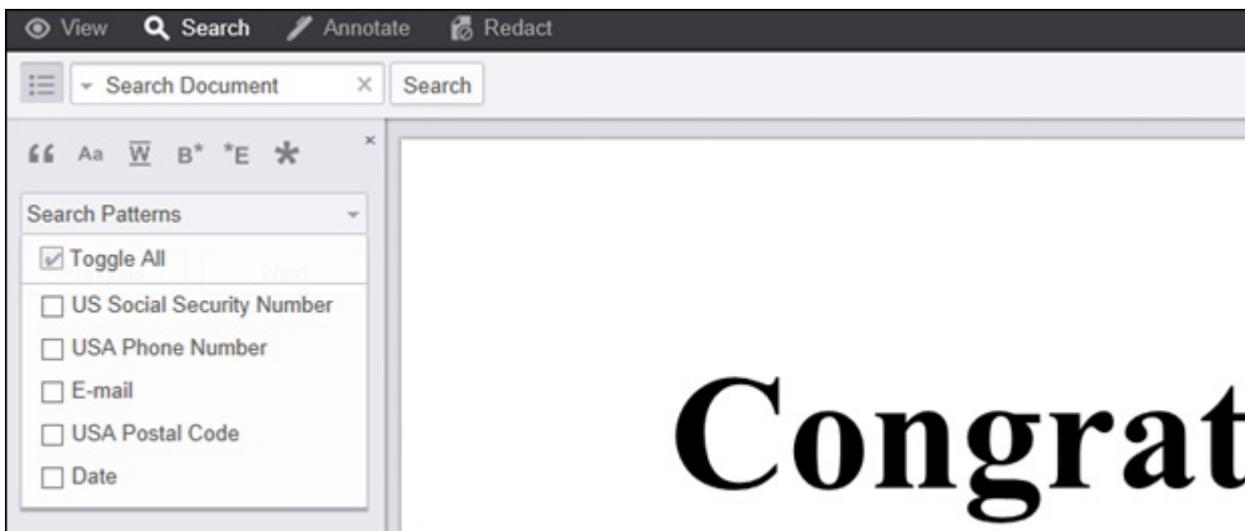
<script type="text/javascript">
var viewingSessionId =
'<%=HttpUtility.JavaScriptStringEncode(viewingSessionId)%>';
//Retrieve the searchJson (search data) into javascript
var searchTerms = <%=searchJson%>;
var pluginOptions = {
  documentID: viewingSessionId,
  predefinedSearch: searchTerms,

```

```

+ (document).ready(function() {
    var viewerControl = $("#viewer1").pccViewer(pluginOptions).viewerControl;
});
</script>

```



## Predefined Search Patterns

Parameter	Data Type	Description
highlightColor	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"
searchOnInit	Boolean	Run search on launch.
globalOptions	Object	Set the default search options for each of the predefined search terms. This is overridden by the term-level "options" parameter.  <b>Example</b> <pre>predefinedSearch : {   globalOptions: {     matchCase: false,     endsWith: false,     beginsWith: false,     matchWholeWord: false   } }</pre>
terms	Array	An array of objects that represent the search terms that will be available in the predefined menu.  <b>Example</b> <pre>predefinedSearch : {</pre>

```

    {
      searchTerm: "llama"
    }
  ]
}

```

## Predefined Search Terms

Parameter	Data Type	Description
searchTerm	String	The search string for the term object. This is overridden by the userDefinedRegex parameter.
searchTermsIsRegex	Boolean	When set to true will use userDefinedRegex to execute the search
userDefinedRegex	String	<p>A regular expression that will be searched in place of searchTerm. 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> <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p><b>Example</b></p> <pre> predefinedSearch : {   terms: [     {       searchTerm: "4 digits"       userdefinedRegex: "(\\d{4})"     }   ] } </pre> </div>
description	String	Description of the search term. If description is not defined "searchTerm" will be used.
highlightColor	String	<p>When specified system will use this value to show the highlight color for this search term. When not specified system will generate a color.</p> <p>Ex: highlightColor: "#FFFF20"</p>

```
options: {  
  "matchCase": false,  
  "endsWith": false,  
  "beginsWith": false,  
  "matchWholeWord": false,  
  "exactPhrase": false  
}
```

## Localizing the Viewer

The Viewer has a language localization feature which allows you to customize labels and text using the **language.json** file. The language.json file is included in all the sample projects. The language object is passed as a parameter in the Viewer plugin configuration options:

### Example

```
var pluginOptions = {  
  documentID: viewingSessionId,  
  language: languageItems,  
  template: htmlTemplates  
}
```

### Editing language.json

The language.json file is in JSON(<http://json.org/>) format. Pay close attention to formatting to ensure that there are no syntax errors. You may want to validate the file using a JSON validator like <http://jsonlint.com/>.

### Using the Language Parameters

In viewer.js the language parameters are used in various places. Here a message is being shown 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.

## Using a Custom Resource Path

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 Title

```
var pluginOptions = {
  documentID: viewingSessionId,
  language: languageItems,
  template: htmlTemplates,
  resourcePath: 'images'
}
```

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);  
}
```

## Adding Custom Image Stamps

You can customize the image stamps available to Viewer users by adding or deleting image files from the ImageStamp folder, located by default in C:\ProgramData\Accusoft\Prizm\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

To add more supported file types, refer to the [PrizmDoc Application Services \(PAS\)](#) configuration options.

### Example

```
imageStamps.validTypes: ["png", "jpg", "jpeg", "gif", "svg", "webp", "tiff",  
"tif"]
```

## Configure PrizmDoc Application Services (PAS)

This section contains the following information:

- [PrizmDoc Application Services \(PAS\) Configuration](#)
- [PrizmDoc Application Services Database Administration & Maintenance](#)
- [How & when to use CORS with PrizmDoc](#)

## PrizmDoc Application Services (PAS) Configuration

### PrizmDoc Application Services Configuration

The Prizm Application Services 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.

### 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` properties.

`port` defines the port that PAS will use to listen to its HTTP connection.

`secretKey` is a value that is used for identification on specific routes (as noted in the PAS API reference). Whenever noted as required in the API, PAS will expect that this value be set to the `Accusoft-Secret` header. To keep your application secure, you are strongly encouraged to change this to a unique string value.

`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. 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.

### Configuring the PrizmDoc Server connection

The connection to your desired PrizmDoc Server, whether Self-Hosted or Accusoft Cloud-Hosted, 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 an Accusoft Cloud-Hosted PrizmDoc Server. This value will be ignored for a self-hosted server.

### Configuring storage

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` - the 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.

*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.*

### Examples:

```
documents.storage: "filesystem"  
documents.path: "/usr/share/prizm/Samples/Documents"
```

```
markupLayerRecords.storage: "filesystem"  
markupLayerRecords.path: "%ALLUSERSPROFILE%\Accusoft\Prizm\MarkupLayerRecords"
```

```
viewingPackagesData.storage: "database"
```

*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

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 `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 preconvert 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.

### Examples:

```
feature.viewingPackages: enabled
```

## Configuring the database

*Note: 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.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.

## 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`.

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.

### Examples:

```
cors.enabled: true
cors.allowedOrigins: [ "http://example.com", "https://example.com" ]
```

## PrizmDoc Application Services Database Administration & Maintenance

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 the 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. You can find out more about this API in the [Viewing Package Validators developer reference](#).

### Product and Database Updates

As the Pre-Conversion 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 will be able to find out more about this in the [PrizmDoc Release Notes](#) in the event that a database migration is required in the future.

## How & when to use CORS with PrizmDoc

### Do you need to use CORS?

We recommend setting up your web-application similarly or having PRIZMDOC Application Services (PAS) proxied behind a web-server, which is discussed in the topic, [Configure PAS in Your Server's Entry Point](#).

## How PrizmDoc supports CORS

While we do not recommend a design that requires CORS, it is possible to configure PAS to handle those requests. Use the following steps to accomplish this:

1. Find your **PAS configuration file**. If you do not know where it is located, refer to the topic, [PrizmDoc Application Services \(PAS\) Configuration](#).
2. Next, you will need to modify the **CORS settings** within your PAS configuration file. For information on those settings, refer to the topic, [PrizmDoc Application Services \(PAS\) Configuration](#).
3. After the changes have been made, restart the **PAS Service** for the changes to take effect. For information on how to start and stop PAS, refer to the topic, [Starting & Stopping the PrizmDoc Application Services](#).
4. 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](#).

 There is no native support for CORS in Internet Explorer 8 and 9.

## Configure the PrizmDoc Server

The PCCIS service can be configured using the pcc.config file for both Windows and Linux. The parameters can be configured for the following services:

- PCCIS

The location of the configuration file:

- Windows: C:\Prizm\PCCIS\ServiceHost
- Linux: /usr/share/prizm/pccis/ServiceHost/

The location of the configuration file on a per-project basis, for example, the full-viewer-sample:

full-viewer-sample\viewer-webtier

 The configuration file will perform environment variable expansion. The environment variable must be enclosed with the % character and must contain upper or lowercase letters or an underscore. For example, %ALLUSERSPROFILE% or %my\_path%.

### Windows & Linux

#### PCCIS

The following options are available for configuration within the PCCIS pcc.config file:

Property	Default Value	Supported Values	Description
DocumentPath	Windows: %ALLUSERSPROFILE%\Accusoft\Prizm\DocumentCache Linux: /usr/share/prizm/cache/DocumentCache/	Any valid path to a directory with read and write permissions.	Part of the PCCIS cache where the original source document files are stored. Read/write access is required for this directory. UNC paths are supported but it is recommended to set this path to a directory on a local drive for best performance. PCCIS will attempt to create this directory if it does not exist.  See How to <a href="#">Implement Caching Strategies</a> for more details.
GroupStateFolder	Windows: %ALLUSERSPROFILE%\Accusoft\Prizm\GroupState Linux: /usr/share/prizm/cache/GroupState/	Any valid path to a directory with read and write permissions.	Part of the PCCIS cache where persistent viewing session data is stored. Read/write access is required for this directory. UNC paths are supported but it is recommended to set path to a directory on a local drive for best performance. PCCIS will attempt to create this directory if it does not exist.  See How to <a href="#">Implement Caching Strategies</a> for more details.
TempcachePath	Windows: %ALLUSERSPROFILE%\Accusoft\Prizm\Cache Linux:	Any valid path to a directory with read and write permissions.	Part of the PCCIS cache where temporary conversion files are stored. Read/write access is required for this directory. UNC paths are supported but it is recommended to set path to a directory on a local drive for best performance. PCCIS will attempt to create this directory

UserDocumentFolder	Windows: %ALLUSERSPROFILE%\Accusoft\Prizm\UserDocuments Linux: /usr/share/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". This directory, when combined with the filename set in the externalId JSON property when creating a new viewing session, provides the full path to the local file. This method for providing source documents to PCCIS is useful if the files already exist on the server that is hosting the service. PCCIS will only read from this directory and copy necessary documents to its own cache for processing. Setting the externalId to a filename with partial path is supported. For example, a value of "\Group1\Document2.pdf" in the externalId JSON property would produce the full path "C:\ProgramData\Accusoft\Prizm\Documents\Group1\Document2.pdf" on Windows.  See <a href="#">How to Transfer Your Document to PrizmDoc Server</a> for more details.
PdfConversionServiceScheme	http	http	The scheme name used by an internal conversion service. Note: This value should not be changed.
PdfConversionServiceHost	localhost	localhost, 127.0.0.1	The host name of the URI used by an internal conversion service. Note: This value should not be changed.
PdfConversionServicePort	38505	Any open HTTP port on the server	The port number used by an internal conversion service. Note: This value should not be changed.
PdfConversionServicePath	PDFCS	PDFCS	The resource path of the URI used by the internal conversion service. Note: This value should not be changed.
RasterConversionServiceScheme	http	http	The scheme name used by an internal conversion service. Note: This value should not be changed.
RasterConversionServiceHost	localhost	localhost, 127.0.0.1	The host name of the URI used by an internal conversion service. Note: This value should not be changed.
RasterConversionServicePort	38502	Any open HTTP port on the server	The port number used by an internal conversion service. Note: This value should not be changed.
RasterConversionServicePath	RCS	Any valid URI resource path string, RCS	The resource path of the URI used by an internal conversion service. Note: This value should not be changed.
VectorConversionServiceScheme	http	http	The scheme name used by an internal conversion service. Note: This value should not be changed.
VectorConversionServiceHost	localhost	localhost, 127.0.0.1	The host name of the URI used by an internal conversion service. Note: This value should not be changed.
VectorConversionServicePort	38508	Any open HTTP port on the server	The port number used by an internal conversion service. Note: This value should not be changed.
VectorConversionServicePath	VCS	VCS	The resource path of the URI used by an internal conversion service. Note: This value should not be changed.
EnableSourceDocumentDownload	true	true, false	When set to true, this property allows the source document of a valid and active viewing session to be downloaded via an HTTP GET request to PCCIS. When false, or any other value beside true, 403 Forbidden is returned.  WARNING: If this property is true, the source document may be downloaded in unencrypted form even if the <EncryptPageContent> flag is set to true.
ViewingSessionTimeout	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.  See How to <a href="#">Implement Caching Strategies</a> for more details.
CacheExpirationPeriod	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.  See How to <a href="#">Implement Caching Strategies</a> for more details.
ViewingSessionIdEncryptionKey	E9rU73lZ2vd0he8Ls/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 an AES encrypted, Base64 encoded value of a string in the format of: <internal ID>/<server's host name>/<Auth-Token header value> <b>Internal ID:</b> This value is a unique GUID that is internally created by PCCIS for each new viewing session. <b>Server's Host Name:</b> Aptly named, this value is the hostname of the server on which PCCIS is running. <b>Auth-Token Header Value:</b> If the "Auth-Token" HTTP header exists in the initial POST request to create a viewing session, its value will be

			<p>which a proxy might need.</p> <p>See <a href="#">PrizmDoc Multi-Server Mode</a> for more details.</p>
ViewingSessionIdEncryptionIv	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.
ViewingSessionPropertyDocumentSource	api,http	api, http, file or any combination of the three separated by a comma.	<p>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.</p> <p>This value can be api, http, file, or a combination of two or more. When combining values, separate them with a comma (,). For example, the value "api,http" would allow the documentSource property to be set to api or http, but not file. 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.</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>
ViewingSessionPropertyExternalId	.*	Valid regular expression using the <a href="#">.NET Regular Expression Language</a> .	<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>
ViewingSessionPropertyDocumentExtension	.*	Valid regular expression using the <a href="#">.NET Regular Expression Language</a> .	<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>
ViewingSessionPropertyCountOfInitialPages	min=0,max=10	A string in the format of "min=<minValue>,max=<maxValue>" where minValue is 0 or a positive integer value and maxValue is 0 or a positive integer value greater than or equal to minValue.	<p>Creates a range filter that will be applied to the "countOfInitialPages" 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>
ViewingSessionPropertyPageContentEncryption	default	enabled, disabled, default, or any (which means either enabled, disabled or default is acceptable)	<p>Creates a range 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>
Html5RenderRasterResolution	min=100,max=300	A string in the format of "min=<minValue>,max=<maxValue>" where minValue is a positive integer value and maxValue is a positive integer value greater than or equal to minValue.	<p>Creates a range filter that will be applied to the "render.html5.rasterResolution" 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>

		either true or false is acceptable)	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.  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.
ViewingSessionPropertyServerCaching	any	none, full, any (which means either none or full is acceptable)	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.  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.  See <a href="#">How to Implement Caching Strategies</a> for more details.
EncryptPageContent	false	true, false	Enables or disables the encryption of page content that is sent to the Viewer from PCCIS. This helps to prevent the unauthorized access of page content over the wire and as it is stored in the browser's cache.  See <a href="#">how to encrypt page content</a> for more details.
PageInteractiveTimeout	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 in error.
DocumentInteractiveTimeout	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 in error.
DocumentAcquisitionTimeout	25000	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 (documentSource viewing session property equals "http"), this is the number of milliseconds that request will wait before timing out.
InternalOperationTimeout	100000	An integer recommended to be between 100000 (100 seconds) and 300000 (5 minutes).	When PCCIS begins the conversion for a document, it has InternalOperationTimeout milliseconds to complete all of the conversion and text extraction operations. If this timeout is reached, the viewing session will be stopped because a valid document was not obtained.
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.  Note that if you stop and restart PCCIS, it will no longer preserve the files and may delete them.

## Central Configuration

PrizmDoc Server can be configured using a central configuration file for both Windows and Linux. To do so, **watchdog.config** must contain a property **paths.central\_config\_file** whose value is a path to the central configuration file relative to the PrizmDoc Server install directory. If this value is not provided, legacy configuration will be used instead.

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.

The following options are available for configuration within the central configuration file:

### Licensing

Property	Default Value	Supported Values	Description
license.solutionName	None (Required)	Valid solution name string	PrizmDoc Server solution name.
license.key	None (Required)	Valid license key string	PrizmDoc Server license key.

Property	Default Value	Supported Values	Description
network.publicPort	None (Required)	Any open HTTP port on the server	The public port the REST API will be available on.
network.internalStartingPort	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.
network.clustering.enabled	false	true, false	Set to true to enable multi-server mode.
network.clustering.clusterPort	-	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.
network.clustering.servers	-	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.

## Security

Property	Default Value	Supported Values	Description
security.aesEncryption.key	"E9rU73lZ2vd0he8Ls/hD8A=="	Base64 encoded value of a byte array representing an AES key with a size of 128, 192 or 256 bits.	<p>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:</p> <p>&lt;internal ID&gt;/&lt;server's host name&gt;/&lt;Auth-Token header value&gt;</p> <p>Internal ID: This value is a unique GUID that is internally created by PCCIS for each new viewing session.</p> <p>Server's Host Name: Aptly named, this value is the hostname of the server on which PCCIS is running.</p> <p>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.</p> <p>See <a href="#">PrizmDoc Multi-Server Mode</a> for more details.</p>
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.

## Logging

Property	Default Value	Supported Values	Description
logging.directory	<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.
logging.daysToKeep	7	Any natural number	<p>Number of daily archives to keep for each log.</p> <p>Note: This parameter does not apply to Native Services, which keeps logs for 7 days, regardless of this setting.</p> <p>Logs with 7 day archives:</p> <ul style="list-style-type: none"> <li>• AutoRedactionService.log</li> <li>• FormatDetectionService.log</li> <li>• HTMLConversionService.log</li> <li>• OfficeConversionService.log</li> <li>• PDFConversionService.log</li> <li>• RasterConversionService.log</li> <li>• VectorConversionService.log</li> </ul>

## 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.

## Work Files

Property	Default Value	Supported Values	Description
workFiles.directory	cache.directory/WorkFileCache	Any valid path to a directory with read and write permissions.	Directory where work files are stored.

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.

See [How to Implement Caching Strategies](#) for more details.

## User Documents

Property	Default Value	Supported Values	Description
userDocuments.directory	<install_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".

## REST API Process Resources

Property	Default Value	Supported Values	Description
processIds.lifetime	"20m"	Formatted Value, see Description	The length of time that a redaction creator, markup burner, or content converter 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.  See <a href="#">How to Implement Caching Strategies</a> for more details.

## 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.  See <a href="#">How to Implement Caching Strategies</a> for more details.
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.  See <a href="#">How to Implement Caching Strategies</a> for more details.
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.
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

			<p>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>
viewing.sessionConstraints.countOfInitialPages.max	10	Any natural number not less than viewing.sessionConstraints.countOfInitialPages.min	<p>Maximum allowed value for the "countOfInitialPages" JSON property when creating a new viewing session.</p> <p>Together with viewing.sessionConstraints.countOfInitialPages.min 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>
viewing.sessionConstraints.documentExtension.regex	".*"	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>
viewing.sessionConstraints.externalId.regex	".*"	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>
viewing.sessionConstraints.pageContentEncryption.allowedValues	["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>
viewing.sessionConstraints.serverCaching.allowedValues	["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>See <a href="#">How to Implement Caching Strategies</a> for more details.</p>
viewing.sessionConstraints.render.alwaysUseRaster.allowedValues	[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.</p>

viewing.sessionConstraints.minSecondsAvailable.max	86400	Any integer greater than 0	server behavior. 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.
--	-------	----------------------------	--

## Fidelity

Property	Default Value	Supported Values	Description
fidelity.svgMaxImageSize	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.
fidelity.vectorBackgroundColor.view.default	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 fidelity.vectorBackgroundColor.view.override to define background color, ignoring the background color specified in the document.
fidelity.vectorBackgroundColor.view.override	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 fidelity.vectorBackgroundColor.view.default.
fidelity.vectorTolerance	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.
fidelity.msOfficeDocumentsRenderer	"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, please contact <a href="#">Accusoft Sales</a> or <a href="#">Customer Support</a> , to obtain the license key with 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 <a href="#">Supported File Formats</a> .
fidelity.msOfficeCluster.host	""	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.
fidelity.msOfficeCluster.port	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 'network.publicPort' parameter of the remote server.  To connect to a load balancer specify 'network.clustering.clusterPort' parameter of the cluster.

## File Types

Property	Default Value	Supported Values	Description
----------	---------------	------------------	-------------

Property	Default Value	Supported Values	Description
fileTypes.office.disableExternalHyperlinks	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.
fileTypes.excel.pagination.enabled	true	true, false	Controls whether or not pagination is enabled for Excel files. If so, the following properties define the min and max page width and height values for pagination.
fileTypes.excel.pagination.dimensions.minWidth	"11.0in"	String consisting of a positive number followed by "in"	Controls the minimum page width for pagination of Excel files if fileTypes.excel.pagination.enabled is set to true.
fileTypes.excel.pagination.dimensions.maxWidth	"22.0in"	String consisting of a positive number followed by "in"	Controls the maximum page width for pagination of Excel files if fileTypes.excel.pagination.enabled is set to true.
fileTypes.excel.pagination.dimensions.minHeight	"8.5in"	String consisting of a positive number followed by "in"	Controls the minimum page height for pagination of Excel files if fileTypes.excel.pagination.enabled is set to true.
fileTypes.excel.pagination.dimensions.maxHeight	"17.0in"	String consisting of a positive number followed by "in"	Controls the maximum page height for pagination of Excel files if fileTypes.excel.pagination.enabled is set to true.
fileTypes.excel.renderGridlines	true	true, false	Specifies whether or not the Excel gridlines in all worksheets of the workbook should be rendered.
fileTypes.excel.renderOnlyPrintArea	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.
fileTypes.excel.renderHeadersAndFooters	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.
fileTypes.excel.renderHiddenContent	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.

## Resource Usage

Property	Default Value	Supported Values	Description
resourceUsage.pccis.instances	3	Any integer greater than 0	The number of PCCIS ASP.NET application instances to run concurrently.
resourceUsage.ocs.numInstances	"auto"	Any integer greater than 0, or the string "auto"	The number of office 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.
resourceUsage.ocs.numThreads	"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.
resourceUsage.ocs.numPorts	"auto"	Any integer greater than 0, or the string "auto"	The number of ports which can be used internally for communication with the office 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.

## Upgrade to Central Configuration

With PrizmDoc v10.5, Central Configuration was introduced to the product. This information in this topic will help you keep the Legacy Configuration in place or upgrade to the Central Configuration:

- **Legacy Configuration** - In older versions of the product, configuration was split over multiple config files. Users who wish to continue using this legacy configuration should remove the **paths.central\_config\_file** property from **watchdog.config**, which will cause the product to ignore the central configuration and use the previously existing config files.
- **Central Configuration** - Users who wish to transfer their configuration settings from the old files to the new central config should consult the following table, which shows the legacy file and property that corresponds to each central config property:

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 multi-server mode, the central

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 multi-server 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	-
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.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	-

## Configuring Ports

### Port Assignment with Central Configuration

If you are using the new central configuration file, port assignment is controlled via three properties:

- **network.publicPort** - The public entry port. The chosen port must be accessible by all servers which need to call the PrizmDoc RESTful APIs.
- **network.clustering.clusterPort** - The port PrizmDoc Servers use to communicate between each other when setup in a cluster. The port must be accessible by all servers in the cluster.
- **network.internalStartingPort** - The first of a series of 200 consecutive ports which the product can use for various internal micro-services which need to communicate between one another via HTTP. For example, when set to 19000, the product will use ports 19000 through 19199 for its own internal purposes. These ports must not be accessible from outside of the server, for security reasons.

### Port Assignment with Legacy Configuration

If you are **not** using the new central configuration file, then port assignment is controlled largely through the `watchdog.config` file:

- **Windows:** `C:\Prizm\PCCIS\Watchdog\watchdog.config`
- **Linux:** `/usr/share/prizm/pccis/Watchdog/watchdog.config`

The two most important ports to assign are the `sep_port` and, if using multi-server mode, the `cep_port`:

- If you are running in **single-server** mode, make sure the port defined by **sep\_port** is accessible to all servers which need to call the PrizmDoc RESTful APIs.
- If you are running in **multi-server** mode, make sure the port defined by **sep\_port** is accessible by all other PrizmDoc Servers running in your cluster, and **cep\_port** is accessible to all servers which need to call the PrizmDoc RESTful APIs.

Many existing services allow you to specify the specific port they run on in the `watchdog.config` file (this is a level of detail we are transitioning away from).

For internal services that don't allow you to control their specific port, the **internal\_starting\_port** property defines a range of 200 consecutive ports which the product may use dynamically for these services. Ports assigned to internal services must not be accessible from outside of the server, for security reasons.

## Format Detection Configuration

PrizmDoc Server rely on the WorkFile for the temporary storage of files so they can be shared by the various services which operate on them. By default, Format Detection is enabled for all files added to the WorkFile service either directly or through viewing sessions. Users may wish to disable format detection to gain a small reduction in processing. Format Detection may be configured by editing the Watchdog configuration file located in the following default directories:

- **Windows:** C:\Prizm\PCCIS\Watchdog\watchdog.config
- **Linux:** /usr/share/prizm/pccis/Watchdog/watchdog.config

The Workfile service configuration provides a Boolean parameter to enable or disable Format Detection:

### Example

```
"workfileService": {  
  ...  
  "formatDetection": {  
    "enabled": true  
  },  
  ...  
},
```

- **enabled** - A value of true indicates that Format Detection will be performed for each file added to the WorkFile Service. A false value will disable Format Detection.

### Enabled Format Detection Behavior

The Format Detection Services can uniquely identify a large number of file types including office documents, imaging formats, CAD formats, and many others. The addition of Format Detection makes the FileExtension parameters in the WorkFile and ViewingSession APIs optional in most cases.

When a document is identified by the Format Detection Service, if no extension is provided, the detected format and file extension will be associated with the document.

In the case where the document cannot be identified by the Format Detection Service, if a file extension was provided, then that extension will be used as a fallback to be associated with the document. If the document cannot be identified and no file extension was provided, an error code will be returned. See the [WorkFile](#) or [ViewingSession](#) API for more details.

### Disabled Format Detection Behavior

When Format Detection is disabled, a file extension must always be provided when adding a document through the WorkFile or ViewingSession APIs.

## Adjust Caching Parameters for PrizmDoc Server

When PrizmDoc Server receives a request to view a new document, also called a viewing session, it

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

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.

### 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.

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:

$$\text{Total Cache Lifetime} = \text{Session Lifetime} + \max(\text{Session Lifetime}, \text{Cache Lifetime})$$

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.

See the [Caching Strategies](#) topic for details and recommendations for the viewing session timeout and cache expiration period values in your application.

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

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, **ViewingSessionTimeout** and **CacheExpirationPeriod**. These parameters are found in the main PrizmDoc Server configuration file located in the following default directories:

- **Linux:** /usr/share/prizm/pccis/ServiceHost/pcc.config
- **Windows:** C:\Prizm\PCCIS\ServiceHost\pcc.config

```
<ViewingSessionTimeout>20m</ViewingSessionTimeout>  
<CacheExpirationPeriod>1d</CacheExpirationPeriod>
```

The viewing session timeout is the length of time that a viewing session remains usable.

The cache expiration period is the length of time that a document is cached and can be potentially reused by other new viewing sessions.

The cache lifetime of a document can be calculated by adding the viewing session timeout value to the maximum of either the viewing session timeout or cache expiration period.

This can be expressed as the following formula:

**Cache Lifetime = Viewing Session Timeout + max(Viewing Session Timeout, Cache Expiration Period)**

## 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.

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

The directories that PrizmDoc Server uses for caching are user-configurable. To change them you will need to set a few different parameters in different configuration files. Mentioned already in the **Cache Lifetime** section above, the main PrizmDoc Server configuration file contains the bulk of the cache location parameters and can be found in the following default directories:

- **Linux:** `/usr/share/prizm/pccis/ServiceHost/pcc.config`
- **Windows:** `C:\Prizm\PCCIS\ServiceHost\pcc.config`

### Example

```
<DocumentPath>/usr/share/prizm/cache/DocumentCache</DocumentPath>  
<GroupStateFolder>/usr/share/prizm/cache/GroupState</GroupStateFolder>  
<TempcachePath>/usr/share/prizm/cache/TempCache</TempcachePath>
```

To be noted:

- Document Path is the location where the original source document files are stored.
- Group State Folder is the location where document metadata and converted content is stored.
- Temp Cache Path is location where intermediate conversion content is stored temporarily.
- Files in the above locations will have a lifetime as discussed in the **Cache Lifetime** section.

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.

## Adjust Office Conversion Settings for Optimal Performance

PrizmDoc Server use an internal component called Office Conversion Service for converting Office documents, such as Word, PowerPoint and Excel.

To customize Office Conversion Service settings, edit the central configuration file. To achieve optimal performance for document conversion, please consider adjusting values for the following parameters:

### Example

```
resourceUsage.ocs.numInstances: 10  
resourceUsage.ocs.numPorts: 16
```

- **resourceUsage.ocs.numInstances** - Represents the number of office document converter processes the service can run. It is recommended to use as many Office conversion instances on the system as there are CPU cores available.
- **resourceUsage.ocs.numThreads** - Number of working threads created by service to handle document processing requests. Recommended value:  $2 * \text{resourceUsage.ocs.numInstances}$ . Setting to 1 or values much smaller than  $\text{resourceUsage.ocs.numInstances}$ , may make the service less responsive. Setting to values greater than  $2 * \text{resourceUsage.ocs.numInstances}$  will not improve performance but will increase resource usage.
- **resourceUsage.ocs.numPorts** - Represents the number of ports which can be used internally for communication with the Office Conversion instances. Recommended value:  $4 * \text{resourceUsage.ocs.numInstances}$ .

### Automatic Settings

The Office Conversion Service can be configured to automatically use optimized values for above mentioned parameters. Just set the value "auto" to the required parameter:

#### Example

```
resourceUsage.ocs.numInstances: auto  
resourceUsage.ocs.numThreads: auto  
resourceUsage.ocs.numPorts: auto
```

The logic for handling "auto" parameters:

- Read `resourceUsage.ocs.numInstances`. If it is "auto", then assign to it the number of virtual CPU cores multiplied by 2 and divided by 3. Otherwise use explicit integer value. For example:
  - On a 4 core machine with hyper-threading that reported 8 virtual cores,  $2/3$  of 8 is 5.3333, which rounds to 5. So 5 Office conversion instances would be used.
  - On a 4 core machine without hyper-threading that reported 4 virtual cores,  $2/3$  of 4 is 2.6666, which rounds to 3. So 3 Office conversion instances would be used.
- Then read `resourceUsage.ocs.numThreads`. If it is "auto", then assign to it `resourceUsage.ocs.numInstances * 2`. Otherwise use explicit integer value.
- Then read `resourceUsage.ocs.numPorts`. If it is "auto", then assign to it `resourceUsage.ocs.numInstances * 4`. Otherwise use explicit integer value.

 The Office Conversion Service performs case sensitive comparison for "auto" value.

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

To customize Office Conversion Service settings, edit the Watchdog configuration file located in the following default directories:

- **Windows:** C:\Prizm\PCCIS\Watchdog\watchdog.config
- **Linux:** /usr/share/prizm/pccis/Watchdog/watchdog.config

To achieve optimal performance for document conversion, please consider adjusting values for the

### Example

```
"officeConversionService": {  
  ...  
  "officeInstanceCount":4,  
  "threadCount":8,  
  "officePortCount":16,  
  ...  
},
```

- **officeInstanceCount** - Represents the number of office document converter processes the service can run. It is recommended to use as many Office conversion instances on the system as there are CPU cores available.
- **threadCount** - Number of working threads created by service to handle document processing requests. Recommended value:  $2 * \text{officeInstanceCount}$ . Setting to 1 or values much smaller than  $\text{officeInstanceCount}$ , may make the service less responsive. Setting to values greater than  $2 * \text{officeInstanceCount}$  will not improve performance but will increase resource usage.
- **officePortCount** - Represents the number of ports which can be used internally for communication with the Office Conversion instances. Recommended value:  $4 * \text{officeInstanceCount}$ .

### Automatic Settings

The Office Conversion Service can be configured to automatically use optimized values for above mentioned parameters. Just set the value "auto" to the required parameter:

### Example

```
"officeConversionService": {  
  ...  
  "officeInstanceCount": "auto",  
  "threadCount": "auto",  
  "officePortCount": "auto",  
  ...  
},
```

The logic for handling "auto" parameters:

- Read `officeInstanceCount`. If it is "auto", then assign to it the number of virtual CPU cores multiplied by 2 and divided by 3. Otherwise use explicit integer value. For example:
  - On a 4 core machine with hyper-threading that reported 8 virtual cores,  $2/3$  of 8 is 5.3333, which rounds to 5. So 5 Office conversion instances would be used.
  - On a 4 core machine without hyper-threading that reported 4 virtual cores,  $2/3$  of 4 is 2.6666, which rounds to 3. So 3 Office conversion instances would be used.
- Then read `threadCount`. If it is "auto", then assign to it  $\text{officeInstanceCount} * 2$ . Otherwise use explicit integer value.
- Then read `officePortCount`. If it is "auto", then assign to it  $\text{officeInstanceCount} * 4$ . Otherwise use explicit integer value.



The Office Conversion Service performs case sensitive comparison for "auto" value.

## Adjust Vector Conversion Settings for Optimal Performance

To customize vector conversion settings, edit the Watchdog configuration file located in the following default directories:

- Windows: C:\Prizm\PCCIS\Watchdog\watchdog.config
- Linux: /usr/share/prizm/pccis/Watchdog/watchdog.config

To achieve optimal performance for document conversion, please consider adjusting values for the following parameters:

### Example

```
"vectorConversionService": {  
    ...  
    "vectorTolerance":0.3,  
    ...  
},
```

- **vectorTolerance** - This parameter lets PCCIS optimize the loading time for vector documents which contain complex polylines, by removing vertices which are very close together (technically, we remove vertices which are very close to the line drawn between their neighbor vertices). The value of the parameter defines the distance in pixels by which points of polylines can be displaced during optimization. The default value is 0.3, which means that polylines will not be displaced farther than 0.3 pixels at 100% zoom. Valid range is from 0.0 to 10.0. A value of 0 means no quality loss, larger values introduce more distortion.

## Change Encryption Keys for Public use Token Generation

As part of the normal operation of the PrizmDoc Back-end RESTful Services, 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 Back-end RESTful Services. 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 Back-end RESTful Services ship 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.

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.
  - If you are using the central configuration, go to Step 2 below.
  - If you are using legacy configuration, go to Step 3 below.

### Step 2: Update the Central Configuration

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.

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

### Step 3: Update the Entry Points Configuration

1. Open the Entry Points config file:
  - **Windows:** C:\Prizm\PCCIS\LoadBalancer\pcc.config
  - **Linux:** /usr/share/prizm/pccis/LoadBalancer/pcc.config
2. Set the **encryptionKey** and **encryptionIv** properties to the Base64 encoded values you created in Step 1.
3. Save and exit the config file.

### Step 4: Update the PCCIS Configuration

1. Open the PCCIS config file:
  - **Windows:** C:\Prizm\PCCIS\ServiceHost\pcc.config
  - **Linux:** /usr/share/prizm/pccis/ServiceHost/pcc.config
2. Set the text within the **ViewingSessionIdEncryptionKey** and **ViewingSessionIdEncryptionIv** XML elements to the Base64 encoded values you created in Step 1.
3. Save and exit the config file.

### Step 5: Update the WorkFile Service Configuration

1. Open the WorkFile Service config file:
  - **Windows:** C:\Prizm\PCCIS\Workfile\workfile.config
  - **Linux:** /usr/share/prizm/pccis/Workfile/workfile.config
2. Set the **affinityTokenKey** and **affinityTokenIv** properties to the Base64 encoded values you created in Step 1.
3. Save and exit the config file.

### Step 6: Update the Redaction Service Configuration

1. Open the Redaction Service config file:
  - **Windows:** C:\Prizm\PCCIS\Redaction\redaction.config

2. Set the **affinityTokenKey** and **affinityTokenIv** properties to the Base64 encoded values you created in Step 1.
3. Save and exit the config file.

### Step 7: Restart PrizmDoc for Changes to Take Effect

- After changing any of the config files above, you need to **restart PrizmDoc**.

## Configure Image Frame Rendering in the PDF Conversion Service

The PDF format specification defines five separate "Page Boundaries" that control various aspects of the imaging process. The PDF Conversion Service supports "Media box" or "Crop box" to convert a source PDF document image(s). The PDF Conversion Service configuration file allows setting what boundary is used. By default, the PDF Conversion Service is configured to use the "Media box" boundary.

To modify the page boundary setting using central configuration:

1. Open the **central config file**.
2. Set the **fileTypes.pdf.pageBoundaries** property value to "**mediabox**" or "**cropbox**".

 The useCropBox property is optional. If the config file does not contain the property, then the service uses the "Media box" boundary for PDF document conversion.

#### Example

```
fileTypes.pdf.pageBoundaries: mediaBox
```

3. Restart **PrizmDoc**.

 The setting of this parameter can affect conversions from PDF to SVG, PNG and JPG.

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

To modify the page boundary setting using legacy configuration:

1. Open the **config file**:
  - **Windows**: C:\Prizm\conf\PDFConversionService.Windows.config
  - **Linux**: /usr/share/prizm/conf/PDFConversionService.Linux.config
2. Set the **useCropBox** property value to **true** to enable "Crop Box" boundary usage. (Or set the property to false to use "Media box".)

 The useCropBox property is optional. If the config file does not contain the property, then the service uses the "Media box" boundary for PDF document conversion.

```
{  
    ...  
    "useCropBox": true,  
    ...  
}
```

3. Restart **PrizmDoc**.

 The Config file "useCropBox" property is supported starting with PrizmDoc v9.1.

## Configure Log File Locations

The PrizmDoc Server are 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 [How to Package Log Files for Product Support](#) to expedite your support request.

The directory where log files are written can be set by the **logging.directory** property in the central configuration file.

### Example

```
logging.directory: /usr/share/prizm/logs
```

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

If you are using legacy configuration, each logging process has a default log file path that you can change if needed. This section will discuss each process one by one and explain how to configure where log files are written.

This section contains the following information:

- [Watchdog](#)
- [Entry Points](#)
- [PCCIS](#)
- [Legacy Log Path Configuration Options](#)

### Watchdog

The Watchdog uses the following default log file paths:

- **Windows:** C:\Prizm\logs\watchdog\*.log
- **Linux:** /usr/share/prizm/logs/watchdog\*.log

The Watchdog will only keep up to 7 days of log files. To change where the Watchdog log files are written:

- **Windows:** C:\Prizm\PCCIS\Watchdog\watchdog.config
  - **Linux:** /usr/share/prizm/pccis/Watchdog/watchdog.config
2. In the paths section, change the **app\_log\_dir property** to the path of a valid directory. Both relative and absolute path are supported but a forward slash should always be used as the directory separator, even on Windows. A relative directory will be relative to the PrizmDoc Server install directory, which is **C:\Prizm\** on Windows and **/usr/share/prizm** on Linux.

## Example

```
"paths": {  
    ...  
    "app_log_dir": [ "C:/some/custom/path" ],  
    ...  
},
```

## Entry Points

The Server and Cloud Entry Points uses the following default log file paths:

- **Windows:** C:\ProgramData\Accusoft\Prizm\Logs\plb.sep\*.log
- **Windows:** C:\ProgramData\Accusoft\Prizm\Logs\plb.cep\*.log
- **Linux:** /usr/share/prizm/logs/plb.sep\*.log
- **Linux:** /usr/share/prizm/logs/plb.cep\*.log

The Entry Points will only keep up to 7 days of log files. To change where the Entry Points log files are written:

1. Open the config file:
  - **Windows:** C:\Prizm\PCCIS\LoadBalancer\pcc.config
  - **Linux:** /usr/share/prizm/pccis/LoadBalancer/pcc.config
2. Change the logFileName property so that the directory part of the existing path is the value you choose. It is recommended that you leave the filename portion of the path as the default value. The new log file directory must be valid and exist.

## Example

```
"logging": {  
    "logFilePath": "%ALLUSERSPROFILE%/Accusoft/Prizm/Logs/plb.log",  
    "level": "info",  
    "count": 7,  
    "period": "1d"  
},
```

## PCCIS

The PCCIS service uses the following default log file paths:

- **Windows:** C:\ProgramData\Accusoft\Prizm\Logs\ImagingServices.log
- **Linux:** /usr/share/prizm/logs/ImagingServices.log

The PCCIS service will only keep up to 7 days of log files. To change where the PCCIS service log file are

1. Open the PCCIS config file:
  - **Windows:** C:\Prizm\PCCIS\ServiceHost\NLog.config
  - **Linux:** /usr/share/prizm/pccis/ServiceHost/NLog.config
2. In the target element, change the fileName property so that the directory part of the existing path is the value you choose. It is recommended that you leave the filename portion of the path as the default value.
3. In the target element, change the archiveFileName property so that the directory part of the existing path is the value you choose. It is recommended that you leave the filename portion of the path as the default value.

### Example

```
<target ... fileName="C:/some/custom/path/ImagingServices.log"  
archiveFileName="C:/some/custom/path/ImagingServices-#{#.log" ... />
```

## Legacy Log Path Configuration Options

Raster Conversion Service, PDF Conversion Service and AutoRedactionService Log Locations can no longer be updated in those respective Configuration files. When using Central Configuration log locations, they are set via the logging.directory parameter. If you choose to bypass the Central Configuration, these values would be set in the watchdog.config:

- Windows: C:\Prizm\PCCIS\Watchdog\watchdog.config
- Linux: /usr/share/prizm/pccis/Watchdog/watchdog.config

## Configure Microsoft Office Conversion Connectivity

PrizmDoc 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.

Please see [Windows Requirements](#) and [Windows Installation](#) sections for information regarding installation of PrizmDoc 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. MSO enabled license is required for the Microsoft Office Conversion connectivity for PrizmDoc Servers running on Linux.
2. Configure following 2 parameters in the [prizm-services-config.yml](#):

**fidelity.msOfficeCluster.host**

PrizmDoc server running on windows to utilize the Microsoft Office Conversion service running on Windows to have native rendering of Microsoft documents in PrizmDoc.

### **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 'network.publicPort' parameter of the remote server to utilize the Microsoft Office Conversion service running on Windows to have native rendering of Microsoft documents in PrizmDoc.

3. Restart PrizmDoc.

 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>
```

## Multi-Server (Cluster) Mode

### Linux Server Configuration

1. MSO enabled license is required for the Microsoft Office Conversion connectivity for PrizmDoc Servers running on Linux.
2. Please configure following 2 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.

### **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.

3. Restart PrizmDoc.

 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 <hostname-of-your-machine>

## File Formats

After performing the configuration steps above, 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 [Supported File Formats](#) section for the information on the exact file formats supported by the Microsoft Office Conversion renderer.

## Customize Excel Pagination Settings for PrizmDoc Server

PrizmDoc Server support two basic pagination modes for rendering Excel documents:

- Non-paginated
- Paginated

In **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. In **paginated** mode, each Excel sheet is divided into a number of pages, such that each page fits into a specific size.

To customize Excel pagination settings, edit the central configuration file. Modify the following parameters to adjust pagination settings:

### Example

```
fileTypes.excel.pagination.enabled: true
fileTypes.excel.pagination.dimensions.minWidth: 11.0in
fileTypes.excel.pagination.dimensions.maxWidth: 22.0in
fileTypes.excel.pagination.dimensions.minHeight: 8.5in
fileTypes.excel.pagination.dimensions.maxHeight: 17.0in
```

- **fileTypes.excel.pagination.enabled** – set to "false" to disable pagination, set to "true" to enable pagination.
- **fileTypes.excel.pagination.minWidth**, **fileTypes.excel.pagination.maxWidth**, **fileTypes.excel.pagination.minHeight**, **fileTypes.excel.pagination.maxHeight** – non-negative doubles. These parameters specify the minimum and maximum sizes, in inches, for pages dynamically generated in the paginated rendering mode. If the 'fileTypes.excel.pagination.enabled' parameter is set to "false", then these 4 parameters are ignored.

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

To customize Excel pagination settings using legacy configuration, edit the Watchdog configuration file

- **Windows:** C:\Prizm\PCCIS\Watchdog\watchdog.config
- **Linux:** /usr/share/prizm/pccis/Watchdog/watchdog.config

Modify the following parameters to adjust pagination settings:

### Example

```
"officeConversionService": {  
    ...  
    "excelPagination": true,  
    "excelPageWidthMin": 11.0,  
    "excelPageWidthMax": 22.0,  
    "excelPageHeightMin": 8.5,  
    "excelPageHeightMax": 17.0  
    ...  
},
```

- **excelPagination** – set to "false" to disable pagination, set to "true" to enable pagination.
- **excelPageWidthMin, excelPageWidthMax, excelPageHeightMin, excelPageHeightMax** – non-negative doubles. These parameters specify the minimum and maximum sizes, in inches, for pages dynamically generated in the paginated rendering mode. If the 'excelPagination' parameter is set to "false", then these 4 parameters are ignored.

## Customize Excel Document View Settings for PrizmDoc Server

PrizmDoc Server supports different view modes for rendering of Excel documents. To customize Excel view settings, edit the central configuration file. Modify the following parameters to adjust Excel viewing settings:

### Example

```
fileTypes.excel.renderGridlines: true  
fileTypes.excel.renderOnlyPrintArea: false  
fileTypes.excel.renderHeadersAndFooters: true  
fileTypes.excel.renderHiddenContent: true
```

- **fileTypes.excel.renderGridlines** – specifies whether or not the Excel gridlines in all worksheets of the workbook should be rendered.
- **fileTypes.excel.renderOnlyPrintArea** – specifies whether the print areas defined in the 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.
- **fileTypes.excel.renderHeadersAndFooters** – 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

- **fileTypes.excel.renderHiddenContent** – specifies whether or not the hidden rows, hidden columns, and whole spreadsheets that are hidden in the original Excel workbook are to be rendered.

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if you are not using the central configuration file.

To customize Excel view settings using legacy configuration, edit the Watchdog configuration file located in the following default directories:

- Windows: C:\Prizm\PCCIS\Watchdog\watchdog.config
- Linux: /usr/share/prizm/pccis/Watchdog/watchdog.config

Modify the following parameters to adjust Excel viewing settings:

### Example

```
"officeConversionService": {  
  ...  
  "excelRenderGridlines": true,  
  "excelRenderOnlyPrintArea": false,  
  "excelRenderHeadersAndFooters": true,  
  "excelRenderHiddenContent": true  
  ...  
},
```

- **excelRenderGridlines** – specifies whether or not the Excel gridlines in all worksheets of the workbook should be rendered.
- **excelRenderOnlyPrintArea** – 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.
- **excelRenderHeadersAndFooters** – 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.
- **excelRenderHiddenContent** – specifies whether or not the hidden rows, hidden columns, and whole spreadsheets that are hidden in the original Excel workbook are to be rendered.

## Customize Text File Encoding for PrizmDoc Server

By default, PrizmDoc Server uses an encoding auto detection mechanism upon loading of text files. This auto detection may fail on some systems or some files due to specific system settings - language, default code page, etc.

To disable encoding auto detection for text files, edit the Watchdog configuration file located in the following default directories:

- Linux: /usr/share/prizm/pccis/Watchdog/watchdog.config

Add following parameter into the "officeConversionService" section:

### Example

```
"officeConversionService": {  
  ...  
  "textFileEncoding": "utf8"  
},
```

Only the "utf8" value is currently supported for the textFileEncoding parameter.

## Disable Excel Pagination for PrizmDoc Server

PrizmDoc Server support two basic pagination modes for rendering Excel documents:

- Non-paginated
- Paginated

In **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. In **paginated** mode, each Excel sheet is divided into a number of pages, such that each page fits into a specific size (see [Customizing Excel Pagination Settings for PrizmDoc Server](#)).

By default, Excel pagination is enabled. To disable pagination, edit the central configuration file. Modify the following parameter to disable or enable pagination:

### Example

```
fileTypes.excel.pagination.enabled: true
```

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

To disable pagination using legacy configuration, edit the Watchdog configuration file located in the following default directories:

- **Windows:** C:\Prizm\PCCIS\Watchdog\watchdog.config
- **Linux:** /usr/share/prizm/pccis/Watchdog/watchdog.config

Modify the following parameter to disable or enable pagination:

### Example

```
"officeConversionService": {  
  ...  
  "excelPagination": true,  
  ...  
}
```

## Implement PrizmDoc Server Caching Strategies

### Why does PrizmDoc Server Cache Files?

The power behind Prizm 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.

### 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.

### Optimizing 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 Tradeoffs

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 PrizmDoc Server sample and

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.

### 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.

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

Set the **GroupStateFolder** setting in the **pcc.config** 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). The other cache folders noted in pcc.config, **DocumentPath** and **TempcachePath**, could benefit too if they were placed onto faster storage devices.

#### Example for Shared Memory Device

```
<GroupStateFolder>/dev/shm/Accusoft/Prizm/GroupState</GroupStateFolder>  
<DocumentPath>/dev/shm/Accusoft/Prizm/DocumentCache</DocumentPath>  
<TempcachePath>/dev/shm/Accusoft/Prizm/Cache</TempcachePath>
```

### Scenario 2:

Viewers are getting errors and the storage device used for the PrizmDoc Server cache is showing errors because the devices are full.

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.

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

The cache expiration period may be specified by **CacheExpirationPeriod** in **pcc.config**. Please note that the time period for **CacheExpirationPeriod** should not be any shorter than the **ViewingSessionTimeout** time period.

If not practical to change device storage device size, try moving the directory specified **TempcachePath** to a different storage device and if that isn't enough do the same for **DocumentPath**. Splitting cache folders to different dedicated storage devices can benefit performance by reducing disk latency for Hard Disk Drives (HDD) compared to having one HDD serving all the viewing sessions.

### Example for Quicker Cache Cleanup

```
<CacheExpirationPeriod>20m</CacheExpirationPeriod>  
<ViewingSessionTimeout>15m</ViewingSessionTimeout>
```

### 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

<ViewingSessionTimeout>1h</ViewingSessionTimeout>

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.

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

The default setting in the **pcc.config** file for **ViewingSessionTimeout** 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

```
<ViewingSessionTimeout>1h</ViewingSessionTimeout>
<CacheExpirationPeriod>1d</CacheExpirationPeriod>
```

### 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

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

## Substitute Fonts for Office Rendering Fidelity

PrizmDoc Server uses a sophisticated logic to select the right font for accurate text rendering within Office documents. 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 out-of-the-box. The configuration file defining the substitution in JSON format is located here:
  - Windows: <install directory>/conf/OfficeConversionService.Font.Windows.config
  - Linux: <install directory>/conf/OfficeConversionService.Font.Linux.config
2. PrizmDoc Server looks for the exact font match within the local PrizmDoc fonts directory containing the fonts installed by the PrizmDoc distribution.
3. PrizmDoc Server looks for the exact font match in the system fonts directory.
4. If none of the steps above solved the substitution, the PrizmDoc Server will dynamically look for the most appropriate font substitution within PrizmDoc 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.

## Start & Stop PrizmDoc Application Services

This section contains the following information:

- [Linux](#)
- [Windows](#)

### Linux

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.

## Example

```
/usr/share/prizm/pas/pm2/pas.sh start
```

## Stop

## Example

```
/usr/share/prizm/pas/pm2/pas.sh stop
```

## Including PAS to the Boot Sequence

You can also configure PAS to be started/stopped together with the system in two steps:

1. Create a symbolic link to the `./pas/pm2/pas.sh` in the `/etc/init.d/` directory:

## Example

```
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.

### RedHat / Cent OS

## Example

```
chkconfig --add pas
```

### Debian / Ubuntu

## Example

```
update-rc.d pas defaults
```

Once done, the system should stop PAS when going to reboot or shutdown, and will be started again when booting the server.



After the first step, you can use the service command to manage PAS instead of invoking the script directly. That way even if you don't need PAS 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 pas start|stop|restart|status
```

## Excluding PAS to 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. It is done as follows:

### RedHat / CentOS

```
chkconfig --del pas
```

Debian / Ubuntu

### Example

```
update-rc.d -f pas remove
```

## 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 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 **PrizmDoc Application Services** 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 PrizmDoc Application Services 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**.

4. Press **Enter**.
5. Expand **Applications and Services Logs**.
6. Click **PrizmDoc Application Services**.

## Start & Stop PrizmDoc Server

This section contains the following information:

- [Linux](#)
- [Windows](#)

### Linux

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
```

#### Including 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.

#### RedHad / CentOS

##### Example

```
chkconfig --add pccis
```

## 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:

### RedHad / CentOS

## Example

```
chkconfig --del pccis
```

### Debian / 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 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.

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**.

## Multi-Server Environments

This section contains the following information:

- **PrizmDoc Server**
  - **Optimize Cache Performance for Multi-Server Mode**

- [Deprecated Configuration Properties](#)
- [PrizmDoc Application Services \(PAS\)](#)
  - [Optimize Cache Performance for Multi-Server Environments](#)
  - [Run PrizmDoc Application Services on Multiple Servers](#)

## PrizmDoc Server

The PrizmDoc Back-end RESTful Services are designed to run out-of-the-box on a single server. In the single-server mode, the PrizmDoc Back-end RESTful 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 Back-end RESTful Services.

If your application requires more bandwidth or processing power than one server can handle, the PrizmDoc Back-end RESTful Services provide a mode that enables request load balancing and routing across multiple servers hosting PrizmDoc Server. The following topic discusses the requirements and considerations for running the PrizmDoc Back-end RESTful Services in multi-server mode.

Additional topics that support multi-server mode:

- [Optimizing Cache Performance for Multi-Server Mode](#)
- [Affinity Tokens & Multi-Server Mode](#)

### Multi-Server Mode

Before getting into the details of multi-server mode, it's important to understand two things about how the PrizmDoc Server generates and serves content:

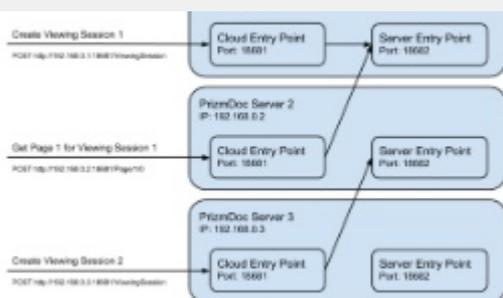
- The majority of requests the PrizmDoc Server handles are centered on document conversions or manipulations. These processes can be computationally expensive, so the PrizmDoc Server attempts to conserve as much CPU resources as possible by caching content as it's created.
- Cached content is only stored locally on the server where it was created and it is directly tied to a specific resource created by the RESTful web service. This means that requests for existing resources must be handled by the PrizmDoc Server on the server that originally created it, otherwise errors will occur.

The caching behavior mentioned above places some requirements on the PrizmDoc Server running in multi-server mode. Luckily, the PrizmDoc Server itself ensures most of these requirements are met.

### How It Works

The multi-server 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 multi-server 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 multi-server 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 CEPs will ensure that the requests are routed to the appropriate machine.

## Configuration

After installation, the PrizmDoc HTTP Service will be running in single-server mode. To enable multi-server mode:

1. Stop the PrizmDoc Server.
2. Open the **central configuration file** in a text editor.
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:

### 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

📄 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.

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**.

### 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 Multi-Server Mode

### Viewing Sessions & Optimizing Cache Performance

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 Services 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 multi-server 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 multi-server 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://192.168.0.1:18682/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 PrizmDoc Application Services API. Refer to the [PrizmDoc Application Services Multi-Server Environments](#) topic for more information on cache optimization for PrizmDoc Application Services.

## Requests for Work Files, Markup Burners, Redaction Creators & Content Converters Require Affinity Tokens

In multi-server 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 multi-server mode, the PrizmDoc Back-end RESTful Services 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://192.168.0.1:18682/PCCIS/V1/WorkFile?FileExtension=pdf
Content-Type: application/octet-stream
[binary data]
200 OK
Content-Type: application/json
{
  "fileId": "5qTYa3gzN9gYUb5SzqUhqg",
  "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://192.168.0.1:18682/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.
```

```
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://192.168.0.1:18682/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": "5ufb3ytUblBxxgSUak_G9Q"
  },
  "affinityToken": "rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w="
}
// 5. Get the burned document once the markup burning process is complete
GET http://192.168.0.1:18682/PCCIS/V1/WorkFile/5ufb3ytUblBxxgSUak_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:

```
// 1. Create first work file, the source PDF document to be converted.
POST http://192.168.0.1:18682/PCCIS/V1/WorkFile?FileExtension=docx
Content-Type: application/octet-stream
[binary data]
200 OK
Content-Type: application/json
{
  "fileId": "5qTYa3gzN9gYUb5SzqUhqg",
  "affinityToken": "rcqmuB9pAa8+4V7fh0lSXzawy/YMQU1g8lLdNDe5l7w="
}
// 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://192.168.0.1:18682/v2/contentConverters
Accusoft-Affinity-Token: rcqmuB9pAa8+4V7fh0lSXzawy/YMQU1g8lLdNDe5l7w=
{
  "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+4V7fh0lSXzawy/YMQU1g8lLdNDe5l7w="
}
// 3. Get status of content conversion process until state is "complete".
GET http://192.168.0.1:18682/v2/contentConverters/E1kNzWtrUJp4rXI5YnLUgw
Accusoft-Affinity-Token: rcqmuB9pAa8+4V7fh0lSXzawy/YMQU1g8lLdNDe5l7w=
200 OK
```

```
"input": {
  "src": {
    "fileId": 5qTYa3gzN9gYUb5SszqUhqg
  },
  "dest": {
    "format": "pdf",
    "pdfOptions": {
      "forceOneFilePerPage": false
    }
  }
},
"expirationDateTime": "2015-12-17T20:38:39.796Z",
"processId": "E1kNzWtrUJp4rXI5YnLUgw",
"state": "complete",
"percentComplete": 100,
"output": {
  "results": [
    {
      "fileId": "KOrSwaqsguevJ97BdmUbXi",
      "src": [{ "fileId": 5qTYa3gzN9gYUb5SszqUhqg, "pages": 1-3" }]
    }
  ]
}
"affinityToken": "rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w="
}
// 5. Get the converted document once the content conversion process is
complete
GET http://192.168.0.1:18682/PCCIS/V1/WorkFile/KOrSwaqsguevJ97BdmUbXi
Accusoft-Affinity-Token: rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g81LdNDe517w=
200 OK
Content-Type: application/pdf
[binary data]
```

## Deprecated Configuration Properties

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

To enable multi-server mode using legacy configuration:

1. Stop the PrizmDoc Server.
2. Open the **Watchdog configuration file** in a text editor. The default path to this file is:
  - **Windows:** C:\Prizm\PCCIS\Watchdog\watchdog.config
  - **Linux:** /usr/share/prizm/pccis/Watchdog/watchdog.config

assigned to valid port numbers.

```
"server_mode": "multi",  
"sep_port": 18681,  
"cep_port": 18682,
```

4. Optionally, set the `cep_servers` value to an array of objects containing address and port values corresponding to each PrizmDoc server on the network node:

### Example

```
"cep_servers": [  
  {  
    "address": "192.168.0.1",  
    "port": 18681  
  },  
  {  
    "address": "192.168.0.2",  
    "port": 18681  
  },  
  {  
    "address": "192.168.0.3",  
    "port": 18681  
  }  
],
```

5. Save and close the **Watchdog configuration file**.
6. Start the PrizmDoc Server.

 If your application makes requests to the PrizmDoc Server 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 Server.

## PrizmDoc Application Services (PAS)

This section contains the following information:

- [Optimize Cache Performance for Multi-Server Environments](#)
- [Run PrizmDoc Application Services on Multiple Servers](#)

## Optimize Cache Performance for Multi-Server Environments

PrizmDoc Application Services (PAS) utilizes cache locations that don't need to be on the same server. If multiple PrizmDoc Application Services 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 PrizmDoc Application Services depending on how it was created. Sessions created using a documentId are stored in PrizmDoc Application Services' 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"
  }
}
```

Note that 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 that if PrizmDoc Server is running in Multi-Server Mode, PrizmDoc Application Services handles the use of affinity hints internally and does not require the user to perform anything additional for optimized PrizmDoc Server cache performance.

## Run PrizmDoc Application Services on Multiple Servers

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

- 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 Self-Hosted or Accusoft Cloud-Hosted entry point.

You can find out more about configuring PrizmDoc Server with a cloud entry point for a multi-server environment in the [PrizmDoc Multi-Server 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.

## Troubleshoot

This section contains the following information:

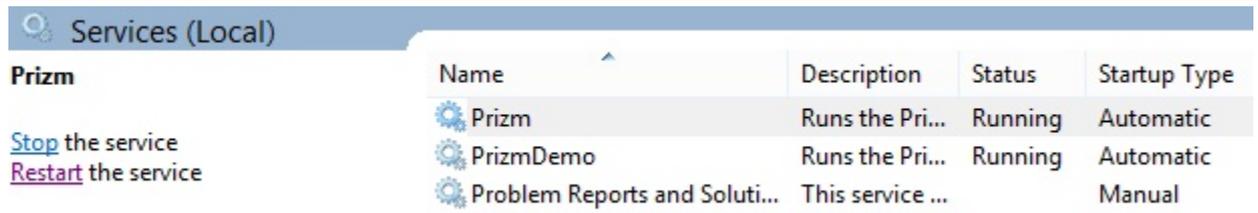
- [Windows](#)
- [Linux](#)
- [Check the System's Health](#)
- [Connection Issues](#)
- [Error Reporting](#)
  - [Getting Started](#)
  - [Accusoft Policy on Log Changes](#)
  - [Search Tips](#)
  - [Package Log Files for Product Support](#)
- [Form Field Detector Error Messages](#)
- [Working Effectively with Large Documents](#)

## Windows

This topic covers troubleshooting tips for PrizmDoc. Note that this topic does not cover issues related to custom web tier implementations.

### The PrizmDoc Viewer isn't Working (PrizmDoc Server Checks)

The following section contains basic troubleshooting steps limited to the PrizmDoc Server itself.

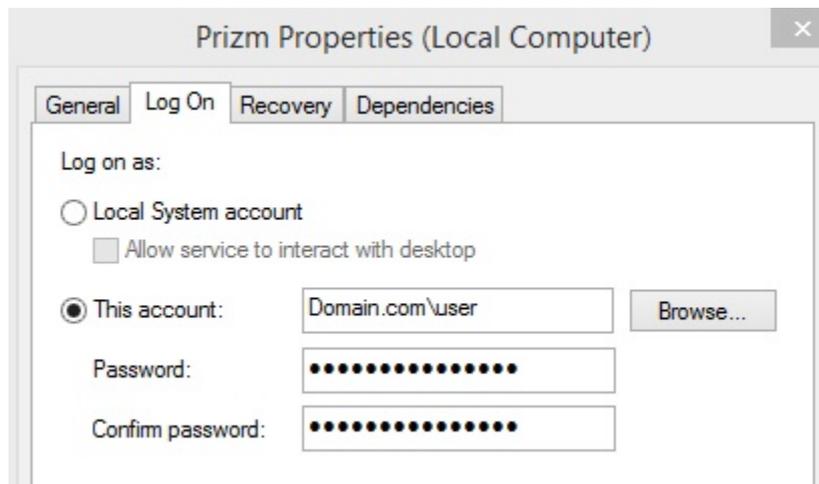


2. Verify that the **PrizmDoc Application Service (PAS)** is running (if your application is using PAS):

The screenshot shows a portion of the Windows Task Manager 'Services' tab. It lists three running services:

Service Name	PID	Company Name	Status
PrizmDemo	2892	PrizmDemo	Running
PrizmApplicationServices	17812	Prizm Application Services	Running
Prizm	5760	Prizm	Running

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**:

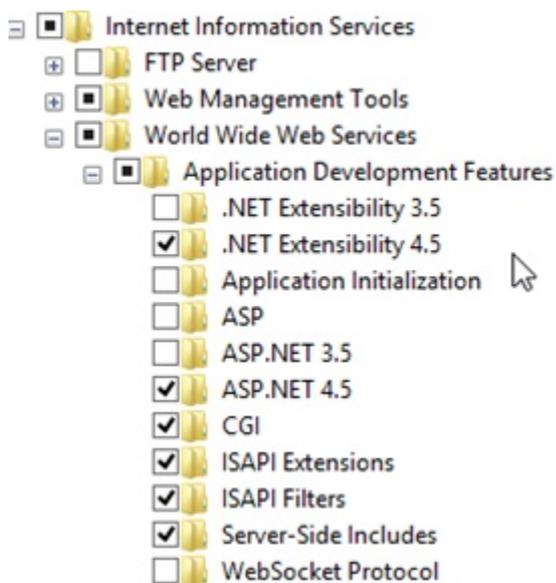


4. Navigate to **C:\ProgramData\Accusoft** and make sure that the user from Step 2 above has full permissions to the folders under **Prizm**:
  - a. Right-click the **Prizm folder** and select **Properties**.
  - b. Navigate to the **Security Tab**.
  - c. Select **Advanced**.
  - d. Select **Change 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.

windows PrizmDoc Service, verify that the password and permissions are set in the same way as Step 3 above.

7. Open the **Windows Features** and check that **.NET Framework 4.5 Advanced Series** is currently installed.

Also, check that the **ASP.NET 4.5 Application development features** are also enabled under **Internet Information Services**:



8. Test a file outside of the Samples. Open a **browser** and navigate to:  
<http://localhost:18681/PCCIS/V1/Static/Viewer/Test>  
Where localhost is a hostname that points to the PrizmDoc Server. Attempt to open a known good file using the Viewer. If the file works, then PrizmDoc is currently functioning. If not, continue troubleshooting.
9. Check the **health** of the PrizmDoc Server:
  - a. If you are using PCC v10.x or later, this can be found at: <http://localhost:18681/admin>  
Your license status will be displayed at the top:

License Status	licensed as 'PCC10'
PCC Version	10.4
OS	Microsoft Windows NT 6.3.9600.0

If you are using PCC v9.x, this can be found at: <http://localhost:18681/PCCIS/V1/Service/Current/Info> (which will display the same data in JSON format).

The health of the services will be displayed.

10. If you find that some of the services are unhealthy (and you have checked permissions, login password of the service user, and are using a known valid license), there are two primary reasons for this:

The system resources for a machine running PrizmDoc are described in [Sizing Servers > PrizmDoc Server](#). 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.

### The cache is corrupted or inaccessible:

Step 3 above is the most common issue for this, however clearing the cache is good practice. To clear the cache manually, use the following steps:

- Stop the **Windows PrizmDoc Server**.
  - Navigate to **C:\Prizm\Cache**
  - Delete the contents of:
    - ContentConversionCache,
    - GroupState,
    - Redactioncache,
    - TempCache,
    - WorkfileCache.
  - Start up the **Windows PrizmDoc Server**, and restart **IIS**.
11. If Steps 1 through 9 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](mailto: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.EAWglWs2X4Aeb3Fes8sU1fA4FfVjYTKfAUA
7
```

Your key will be everything after the equal sign on **Line 6**. Copy and send your **full key** to Accusoft Support.

12. Check that **PAS** is able to connect to the PrizmDoc Server:

#### Example

```
GET http://localhost:3000/servicesConnection
```

Returns the status of PAS connectivity to the PrizmDoc Server, whether local or configured through Accusoft Services.

#### Successful Response:

##### Example

```
200 OK
OK
```

 This response represents 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 those services, please make a call to them directly.

#### Error Response:

580

 The response represents that PAS is not properly configured to communicate with PrizmDoc Server.

13. If everything seems to be functioning, however a file still cannot be shown 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](mailto:support@accusoft.com) for evaluation and submission to our engineering team.

## Linux

This topic covers troubleshooting tips for PrizmDoc. Note that this topic does not cover issues related to custom web tier implementations.

### The PrizmDoc Viewer isn't Working (PrizmDoc Server Checks)

The following section contains basic troubleshooting steps limited to the PrizmDoc Server itself.

1. Verify that the **PrizmDoc Server** is running. Check the **health** of the PrizmDoc Server:
  - a. If you are using PCC v10.x or later, this can be found at: <http://localhost:18681/admin>  
Your license status will be displayed at the top:

License Status	licensed as 'PrizmDoc12'
PCC Version	12.2
OS	Unix 3.13.0.36

- b. If you are using PCC v9.x, this can be found at: <http://localhost:18681/PCCIS/V1/Service/Current/Info> (which will display the same data in JSON format).

 Note that the "License Status" and "PCC Version" may differ on your system from the example above.

Note that "OS" will be your version of Linux.

The health of the services will be displayed.

2. Test a file outside of the Samples. Open a **browser** and navigate to:  
<http://localhost:18681/PCCIS/V1/Static/Viewer/Test>  
Where localhost is a hostname that points to the PrizmDoc Server. Attempt to open a known good file using the Viewer. If the file works, then PrizmDoc is currently functioning. If not, continue troubleshooting..
3. If you find that some of the services are unhealthy (and you have checked permissions, login password of the service user, and are using a known valid license), there are two primary reasons for this:

The system resources for a machine running PrizmDoc are described in [Sizing Servers > PrizmDoc Server](#). 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.

### 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:

- Run: `/usr/share/prizm/scripts/pccis.sh stop`
- Delete the contents of: `/usr/share/prizm/cache/`
- Start up PrizmDoc Server: `/usr/share/prizm/scripts/pccis.sh start`

4. Another common issue is that either PrizmDoc Application Services (PAS) or PrizmDoc Server isn't running.

To check that PAS is running:

- Navigate to: `/usr/share/prizm/pas/pm2`
- Run: `./pas.sh status`

To check that the PrizmDoc Server is running:

- Navigate to: `/usr/share/prizm/scripts`
- Run: `./pccis.sh status`

5. If Steps 1 through 4 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](mailto: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 that your solutionName and license key may differ from the example above.

6. If everything seems to be functioning, however a file still cannot be shown 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](mailto:support@accusoft.com) for evaluation and submission to our engineering team.

## Check the System's Health

become unhealthy, in which case you must restart PrizmDoc (typically by rebooting the server).

To check whether or not a server is healthy, simply send an HTTP **GET** request to **/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 has just started, HTTP 500 may be returned for a short time until the system has completely started up.

### Multi-Server Mode: Use the CEP

If you are running in multi-server mode, send your request to the Cloud Entry Point (CEP), typically running on port 18682:

#### Example

```
GET server:18682/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:

#### Example

```
GET server:18681/PCCIS/V1/Service/Current/Health
```

### HTTP Response

- **HTTP 200** - Server is healthy and running normally.
- **HTTP 500** - Server is unhealthy and must be restarted (or PrizmDoc is still starting up).

### Additional Information

Refer to the API topic, [Health Status](#) for more information on checking the health of PrizmDoc Server.

## Connection Issues

### Accusoft Cloud-Hosted Services

#### Troubleshooting Connection Issues

If there is an issue with your connection, there are a few steps you can take to troubleshoot:

1. Locate your **PrizmDoc Application Services 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):
  - pccServer.hostName: "api.accusoft.com"
  - pccServer.port: 443
  - pccServer.scheme: "https"

to [Starting & Stopping PrizmDoc Application Services](#) for instructions, and try again.

If you are still unable to verify your connection to the Accusoft Cloud-Hosted Services, please contact [Accusoft Support](#).

## Self-Hosted Server

### Troubleshooting Connection Issues

If there is an issue with your connection, there are a few steps you can take to troubleshoot:

1. Locate your **PrizmDoc Application Services 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 single-server mode and 18682 by default for **multi-server mode**. If the settings are not correct, update them and try again.
3. If the values for these parameters are correct, try restarting **PrizmDoc Application Services**. Refer to [Starting & Stopping PrizmDoc Application Services](#) for instructions, and try again.

If you are still unable to verify your connection to the Self-Hosted Server, please contact [Accusoft Support](#).

## Error Reporting

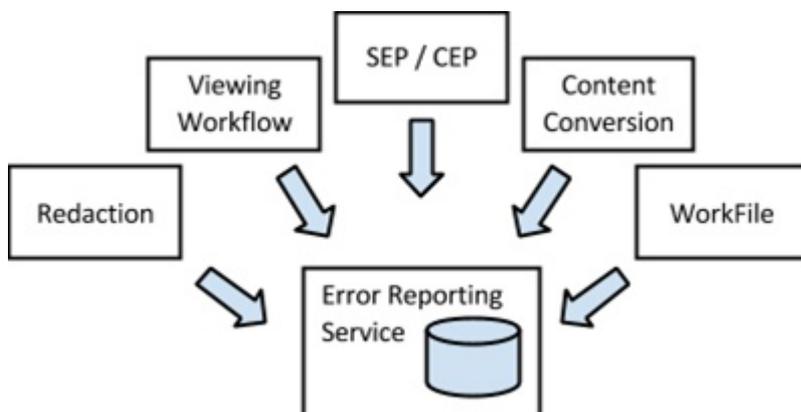
This section contains the following information:

- [Getting Started](#)
- [Accusoft Policy on Log Changes](#)
- [Search Tips](#)
- [Package Log Files for Product Support](#)

## Getting Started

### Overview

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



## Error Reporting Configuration

Errors are reported to a log file located in the directory containing the other service log files. By default this is located on Linux at:

```
/usr/share/prizm/logs/PccErrors.log
```

and on Windows at:

```
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 following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

If not using the central configuration file, then by default the error log is located on Linux at:

```
/usr/share/prizm/logs/PccErrors.log
```

and on Windows at:

```
%ALLUSERSPROFILE%/Accusoft/Prizm/Logs/PccErrors.log
```

These locations, as well as the file rotate and retained archive configurations, can be found in the Watchdog configuration file located under the "errorReportingService" section on Linux at:

```
/usr/share/prizm/pccis/Watchdog/watchdog.config
```

and on Windows at:

```
%ALLUSERSPROFILE%/Accusoft/Prizm/pccis/Watchdog/watchdog.config
```

An example of the Error Reporting Service configuration within the Watchdog configuration file on Windows is shown below:

### Example

```
"errorReportingService": {
```

```
errorLogging : {
  "logFilePath": "%ALLUSERSPROFILE%/Accusoft/Prizm/Logs/PccErrors.log",
  "count": 7,
  "rotateOn": "period",
  "period": "1d",
  "size": "200M"
},
...
}
```

The configuration properties are described below:

<b>logFilePath</b>	This is the path to file to which errors will be reported.
<b>count</b>	The is the maximum number of archive logs to retain when rotating based on period or file size. The default value is 7 which indicates that after 7 file rotations there will be 7 archives (not including the current active log). After the 8th file rotation, the oldest file will be deleted and replaced by the next oldest file. If a value of 0 is used, then at rotation time, the current active log file will be deleted and replaced with a new current active log file.
<b>rotateOn</b>	This indicates whether to base the file rotation on time or size. Its value may be either 'period' or 'size'.
<b>period</b>	When rotateOn is set to 'period' this value indicates the period an error log file will be active until rotated. This property must be a number with a single character unit which may be <b>(y)</b> ears, <b>(m)</b> onths, <b>(w)</b> eeks, <b>(d)</b> ays or <b>(h)</b> ours. By default the period is set to '1d' indicating the file will be rotated once per day at 00:00.
<b>size</b>	When rotateOn is set to 'size' this value indicates the maximum allowed size for the error log file. When a log entry results in a file which exceeds this size, the archive is rotated. This property must be a number with a single character unit which may be <b>(K)</b> ilobytes, <b>(M)</b> egabytes, <b>(G)</b> igabytes, or <b>(T)</b> erabytes. By default the size is set to '200M' indicating that the file will be rotated when it exceeds 209,715,200 bytes.

## Error Report Entries

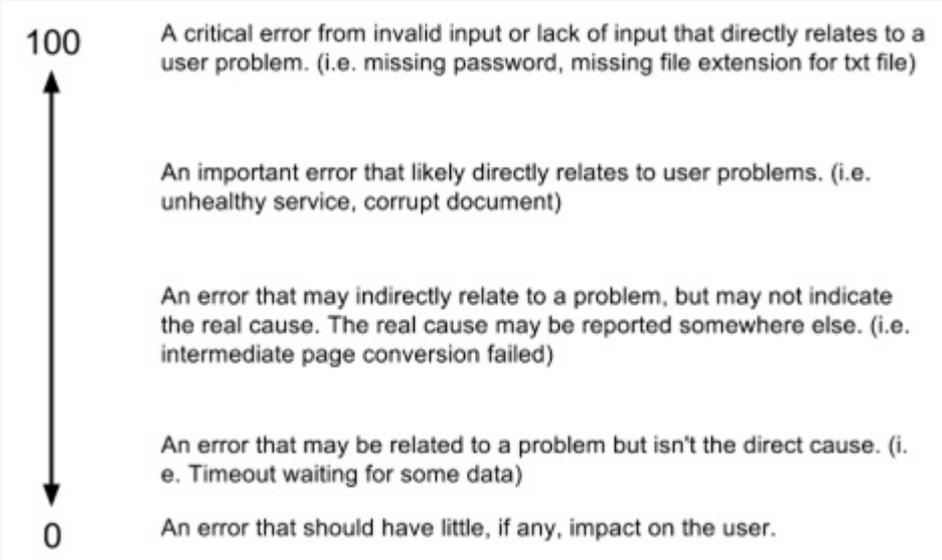
The error log file may contain two type of entries: Errors and Relations. Errors entries describe a specific error event reported by a service and can be used to diagnose issues with a document or service. Relations describe the relationship between two resources and are used to help diagnose an error 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": "
```

```
{  
  "relevance": 100,  
  "errorCode": "DocumentRequiresAPassword",  
  "time": "2015-04-28 20:27:25.0473",  
  "errorId": "f00hLsu_V8TimZm88w1b6w"  
}
```

<b>gid</b>	This is the Global ID assigned to each new request. It uniquely identifies operations resulting from the original request.
<b>service</b>	This is the name of the service which reported the error.
<b>resourceType</b>	This describes the resource related to the error. Examples include ContentConverter, WorkFile, MarkupBurner, RedactionCreator.
<b>resourceId</b>	This is the unique resource identifier (i.e. ViewingSessionId).
<b>relevance</b>	This indicates the importance of the error:  <ul style="list-style-type: none"><li>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)</li><li>An important error that likely directly relates to user problems. (i.e. unhealthy service, corrupt document)</li><li>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)</li><li>An error that may be related to a problem but isn't the direct cause. (i.e. Timeout waiting for some data)</li><li>0: An error that should have little, if any, impact on the user.</li></ul>
<b>errorCode</b>	This is the error code reported by the service. This will be a PascalCased string briefly describing the error.
<b>time</b>	This is the time at which the error was reported.
<b>errorId</b>	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",  
}
```

```
resource":  
  "relation": "RedirectedViewingSession",  
  "relationResourceId":  
    "gRrJ8ay0jV6wBXiqMjxmB4epUrsd7KqVdtsD_BtwAZbhYBVpb4P2ksm0_kcByugmA",  
  "relationId": "EjA6CpDhFuTMP3sTAvMhyA"  
}
```

<b>gid</b>	This is the Global ID assigned to each new request. It uniquely identifies operations resulting from the original request.
<b>service</b>	This is the name of the service which reported the relation.
<b>resourceType</b>	This describes the resource associated with the relation. Examples include ContentConverter, WorkFile, MarkupBurner, RedactionCreator.
<b>resourceId</b>	This is the unique resource identifier (i.e. ViewingSessionId).
<b>relation</b>	This describes the related resource. (i.e. RedirectedViewingSession).
<b>relationResourceId</b>	This is the unique related resource identifier (i.e. ViewingSessionId).
<b>relationId</b>	This is a unique relationId assigned to the error by the Error Reporting Service.

## Accusoft Policy on Log Changes

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

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

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 "sugq5PRxDV4ERAbAQLgpRzKjbKrHp868m6zN2QG-wBFHO3ZX-SlhJ3GIJA8FK4WZzB2DjiJ_gZWTAUdyKeqBcw"
C:\ProgramData\Accusoft\Prizm\Logs\PccErrors.log
```

### On Linux Bash:

```
$> grep sugq5PRxDV4ERAbAQLgpRzKjbKrHp868m6zN2QG-wBFHO3ZX-SlhJ3GIJA8FK4WZzB2DjiJ_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":
"sugq5PRxDV4ERAbAQLgpRzKjbKrHp868m6zN2QG-wBFHO3ZX-SlhJ3GIJA8FK4WZzB2DjiJ_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":
```

```
{ "gid": "13CbzVdHZ/wqVKLCZmfq9A", "time": "2015-05-14 19:26:25.9834", "resourceType": "ViewingSession", "resourceId": "sugq5PRxDV4ERAbAQlGpRzKjBKrHp868m6zN2QG-wBFHO3ZX-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": "nXI4pNPhXmS2Idb8vui1LQ" }
```

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".

## Package Log Files for Product Support

### Linux

With a default installation of PrizmDoc, 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 log files to Accusoft Support, simply create a gzipped tarball containing everything in that directory.

 If your packaged files total less than 25 MB, you can email it to [support@accusoft.com](mailto:support@accusoft.com). However, if your packaged files are larger than 25 MB, you will need to upload it to our FTP site (see <http://www.accusoft.com/faqs/large-files-need-upload-ftp-location-can-provide-upload-files/>).

### Windows

With a default installation of PrizmDoc, 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 log files to Accusoft Support, simply create one or more zip files containing everything in these two directories.

 If your packaged files total less than 25 MB, you can email it to [support@accusoft.com](mailto:support@accusoft.com). However, if your packaged files are larger than 25 MB, you will need to upload it to our FTP site (see <http://www.accusoft.com/faqs/large-files-need-upload-ftp-location-can-provide-upload-files/>).

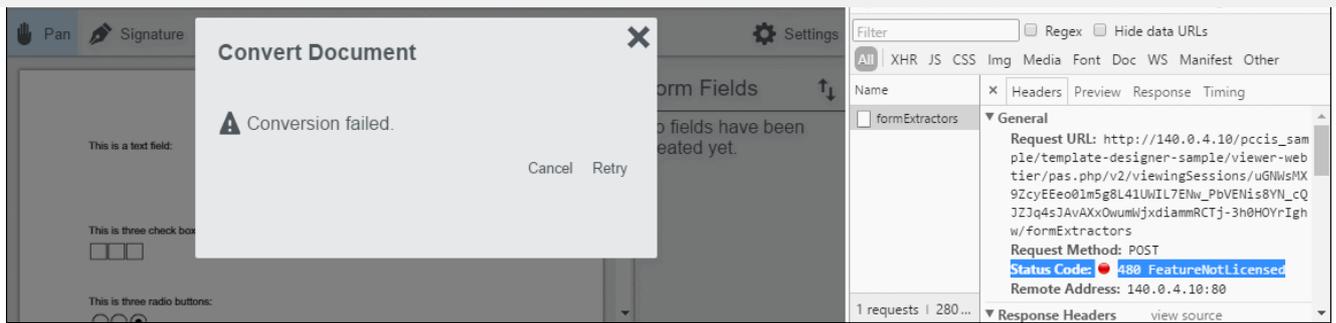
## Form Field Detector Error Messages

To distinguish between a generic Form Field Detector error and when the feature is not licensed, open **Developer tools** in your browser, select the **Network** tab and click the **Retry** button. If your license key doesn't include the Form Field Detector feature, you will see the following Status Code:

### Example

Status Code: 480 FeatureNotLicensed

with the following JSON body: {errorCode:"FeatureNotLicensed"} as shown below:



If you want to enable the feature, refer to the topic [Overview of PrizmDoc 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.

## Developer Guide

This section contains the following information:

- [Introduction](#)
- [Work Effectively with Large Documents](#)
- [Customize the Viewer](#)
  - [Architecture & Design](#)
    - [Design Basics](#)
    - [Responsive Layouts](#)
    - [Architecture Basics](#)
  - [Integrate PrizmDoc Releases with Your Code](#)
  - [Customization Examples](#)
    - [Add a Custom Button](#)
    - [Add Keyboard Shortcuts](#)
    - [Build a Custom User Interface](#)
    - [Change Annotation Default Values](#)
    - [Change the Position of the Menu Bar](#)
    - [Create a Custom Mouse Tool](#)
    - [Create a Custom Tab](#)
    - [Customize the Markup](#)
    - [Customize the Styles](#)
    - [Disable the Print Button](#)
    - [Scroll the Viewer Programmatically](#)
    - [Work with Annotations Programmatically](#)
    - [Reorganize Menus](#)
    - [Set the Initial Zoom Factor](#)
    - [Subscribe to Events](#)
  - [Mouse Tools](#)
  - [Modify viewer.js](#)
  - [Annotation Layers](#)
    - [Load Annotations from the Web Tier](#)
- [PrizmDoc Application Services](#)
  - [Set up Your Database for use with PrizmDoc Application Services](#)
  - [Migrate to PrizmDoc Application Services](#)
  - [Handle Specific Routes from PrizmDoc Application Services using Custom Logic](#)
  - [Pre-Convert Documents](#)
  - [Viewing Packages](#)
- [PrizmDoc Server](#)
  - [Use the PrizmDoc Server API](#)
  - [Markup Burner XML Specification](#)

- 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 Accusoft Cloud-Hosted Servers to Self-Hosted Servers
  - Natively Render Microsoft Office Documents
  - Perform Auto-Redaction
  - Pre-Populate Fields in the E-Signature Viewer
  - Set up a Viewing Session for a CAD Drawing which has XREF Dependencies
  - Transfer Your Document to PrizmDoc Server
  - Use a Viewing Session
  - Watermark Content in a Viewing Session
- Customize the E-Signature Viewers
  - Viewer Modular Design
  - Configure the E-Signature Viewers
  - Build the E-Signature Viewers
  - Fill in Fields Programmatically

## Introduction

PrizmDoc was designed with ultimate flexibility for developers. With a completely customizable user interface and a powerful set of APIs for PrizmDoc Server, we provide you with numerous options to ensure that PrizmDoc meets your needs.

### 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. Many users will be able to integrate the Viewer into their web application with little or no customization needed.

#### Customizing through Configuration

For developers who need minor modifications to the Viewer, customizing through configuration is the simplest option. The Viewer can be customized by modifying the `uiElements` to control which tabs are displayed as well as several other useful customizations.

For example, you may want to disable the user's 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

### 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 the developer 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, a developer 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, the developer will incur some overhead merging customizations with future versions of [PrizmDoc](#).

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 the developer to focus on areas of the Viewer needing modification. Simple customizations such as rearranging, removing or renaming tabs can be done very quickly by modifying the main template (viewerTemplate.html). From there, the developer 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 so the developer can edit the business logic and behavior of the Viewer. For more information, refer to the topic: [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.

### PrizmDoc Application Services

PrizmDoc Application Services abstracts much of the code that you previously had to write or integrate from our samples into a node.js service with a public API. The sample code that ships with PrizmDoc may be all you need to integrate PrizmDoc into your web tier.

However, for more advanced operations, such as managing your annotations in your own database, you may need to interact with the PrizmDoc Application Services API. In addition, PrizmDoc now supports an API for pre-converting documents and centralizing cached data. This is also managed in the PrizmDoc Application Services API. For more information, refer to the [PrizmDoc Application Services](#) section in the Developer Guide.

PrizmDoc Server provides a set of APIs to allow you to interact directly with the services outside of the Viewer. Using the RESTful APIs, you can work programmatically with viewing sessions, manage redaction burning, and check system health. This can all be performed via the API without having to use the Viewer. Refer to the [RESTful API documentation](#) for more details.

You can also convert documents from source files into PDF or TIFF, combine multiple documents and pages of documents into a single new document using Content Conversion Services (CCS). Refer to the [CCS API documentation](#) for more details.

## Advanced Configuration

Developers that are interested in the additional configuration options for PrizmDoc, refer to [Administering PrizmDoc > System Configuration](#).

# Work Effectively with Large Documents

This section gives high-level guidelines to achieve fast end-user interaction with source documents which 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 the PrizmDoc Application Services 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 the PrizmDoc Application Services (that is, you are communicating directly to the back-end PrizmDoc Server to create your viewing sessions), consider adding PrizmDoc Application Services to your environment to take advantage of the [Viewing Packages](#) feature.

If you are not familiar with how the browser, your web tier, PrizmDoc Application Services, and the back-end PrizmDoc Server work together, refer to the [Understanding PrizmDoc](#) overview.

## Use Server-side Search to View Large Documents

### Client-side vs. Server-side Search

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 less 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.

`searchMethodPageCountInresoid` 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).

## Customize the Viewer

This section contains the following information:

- [Architecture & Design](#)
  - [Design Basics](#)
  - [Responsive Layouts](#)
  - [Architecture Basics](#)
- [Integrate PrizmDoc Releases with Your Code](#)
- [Customization Examples](#)
  - [Add a Custom Button](#)
  - [Add Keyboard Shortcuts](#)
  - [Build a Custom User Interface](#)
  - [Change Annotation Default Values](#)
  - [Change the Position of the Menu Bar](#)
  - [Create a Custom Mouse Tool](#)
  - [Create a Custom Tab](#)
  - [Customize the Markup](#)
  - [Customize the Styles](#)
  - [Disable the Print Button](#)
  - [Scroll the Viewer Programmatically](#)
  - [Work with Annotations Programmatically](#)
  - [Reorganize Menus](#)
  - [Set the Initial Zoom Factor](#)
  - [Subscribe to Events](#)
- [Mouse Tools](#)
- [Modify viewer.js](#)
- [Annotation Layers](#)
  - [Load Annotations from the Web Tier](#)

## Architecture & Design

The Viewer offers the following features out of the box:

- A responsive UI
- A jQuery plugin for embedding the full Viewer

- 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 configuration options allow, all UI code is open-source and can be modified to suit 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 separation of concerns.

## API

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

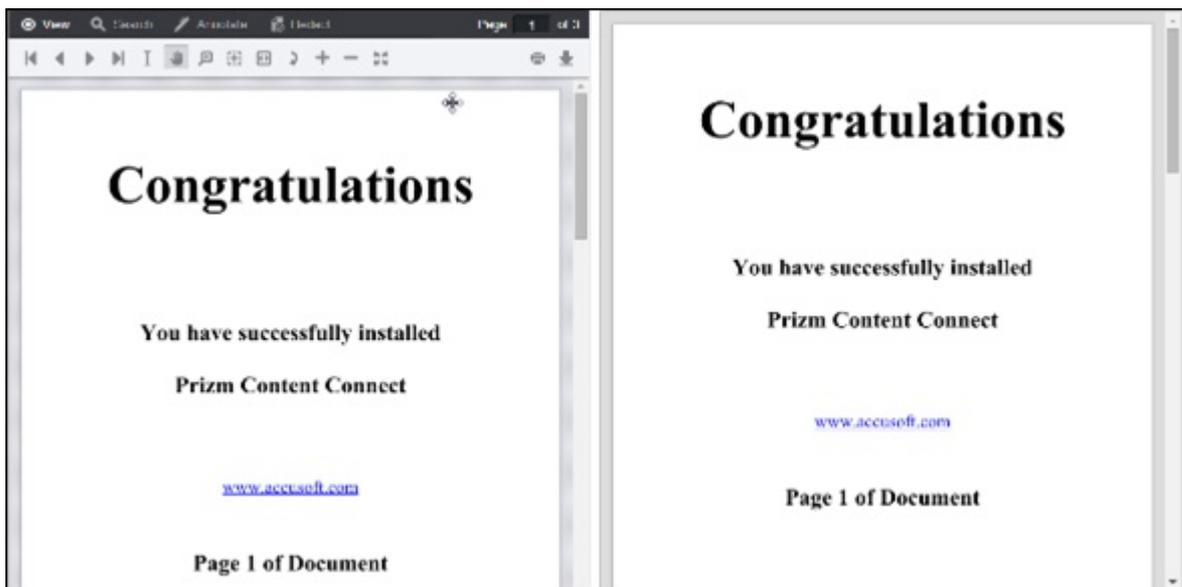
```
var api = $("#myDiv").pccViewer(pluginOptions).viewerControl;  
api.on("PageCountReady", function() {  
    api.changeToLastPage();  
});
```

## 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:

The ViewerControl does not have dependencies on third party libraries.

## Design Basics

### Design Basics

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 utilizes 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

## Polyfills

There are a few polyfills used to provide support for modern 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 these components are in viewer.css with exception to Page List, those styles are defined in

## 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

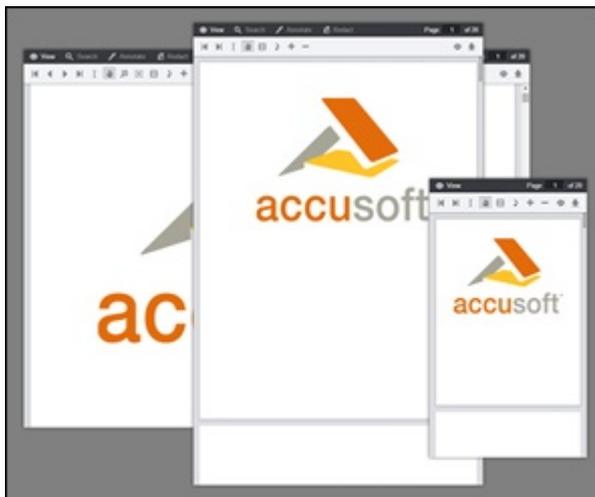
To disable one of the navigation tabs you could comment out the HTML in the templates or pass one of the following configuration parameters to the jQuery viewer plugin:

### Example

```
var pluginOptions = {
  uiElements: {
    redactTab: false
  }
};
```

## Responsive Layouts

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 utilizes CSS3 Media Queries (<http://www.w3.org/TR/css3-mediaqueries/>) with expressions

## 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) {}
```

## 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-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>
```

## 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.

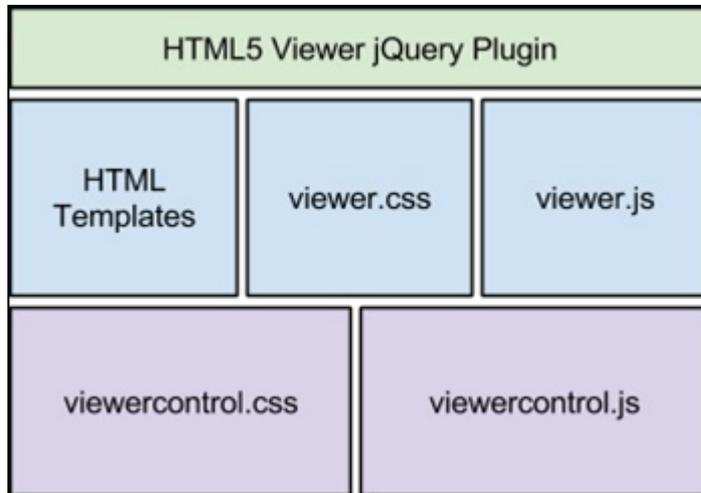
An understanding of the Viewer architecture is not required for the out-of-the box experience offered by the jQuery plugin.

### Beyond the jQuery Plugin

- The jQuery plugin is built using several open-source CSS, JavaScript, and HTML template files.

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.

The implementation of the ViewerControl is in `viewercontrol.js` and `viewercontrol.css`. Unlike UI files, `viewercontrol.js` is closed-source and modification is not supported. However, `viewercontrol.css` can be modified to suit customization needs.



## ViewerControl is Reusable

The ViewerControl is a reusable component, it 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.

### Example

```
var viewerControl =  
    $("#myDiv").pccViewer(pluginOptions).viewerControl;  
  
viewerControl.on("PageCountReady", function() {  
    viewerControl.changeToLastPage();  
});
```

## 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.

### **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.

## Customization Examples

The Viewer offers several ways to customize the Viewer, 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 allowing the developer to edit markup and css. All Viewer functionality is built on top of the Viewer API, allowing complete control over all Viewer functions.

### Configuration Options

parameters 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 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](#)

This section contains a variety of customization examples:

- [Add a Custom Button](#)
- [Add Keyboard Shortcuts](#)
- [Build a Custom User Interface](#)
- [Change Annotation Default Values](#)
- [Change the Position of the Menu Bar](#)
- [Create a Custom Mouse Tool](#)
- [Create a Custom Tab](#)
- [Customize the Markup](#)

- [Disable the Print Button](#)
- [Scroll the Viewer Programmatically](#)
- [Work with Annotations Programmatically](#)
- [Reorganize Menus](#)
- [Set the Initial Zoom Factor](#)
- [Subscribe to Events](#)

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

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.

 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>
```

You've added the button in bold to the rest of the list, to appear first in the right-hand side buttons.

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 **bindMarkp**, 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

First, add a new custom icon to the CSS vocabulary. To support all browsers and screen types, you will

icons already created, named **check.png** and **check@2x.png**, respectively and the icons are available in the images folder.

 The Viewer uses an icon sprite in order to optimize icon image loading. In this example, you will not be using the sprite, and instead include standalone images.

Toward the top of the **viewer.css** file, there are the icon definitions, in the form of **.pccv .pcc-icon-....** You will add the new icon at the end of this section:

## Example

```
.pccv .custom-icon-check { background-image: url(../img/check.png); }
```

Note that to support IPS screens, you will need to add a second icon inside the **@media (min-width:0)** media query. This media query targets all modern browsers capable of scaling icons, to ensure that any high density screen attached to a browser (that can scale icons) will get high resolution images. Here, you will add out 2x icon image:

## Example

```
.pccv .custom-icon-check {  
  background-image: url(../img/check@2x.png);  
  background-size: 26px;  
  background-position: center;  
}
```

Finally, update your HTML markup to use this new icon instead of text:

## Example

```
<button class="myCustomApprovedButton pcc-icon custom-icon-check"></button>
```

You have completed adding a custom button.

## Add Keyboard Shortcuts

### Customizing Keyboard Support in the Viewer

The Viewer includes support for some keyboard shortcuts. This topic will walk through how the [jQuery.hotkeys](https://github.com/jeresig/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.

- [Keyboard Key Combinations for Page Navigation](#)
- [Zoom in / Zoom out](#)
- [Delete Selected Marks](#)
- [Modal Dialogs](#)
- [Adding Keyboard Support without using jQuery.hotkeys Plugin](#)
- [Example Code for using pageup and pagedown Keys](#)

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

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. This file must be referenced in the Viewer's start page( default.aspx for .NET sample, index.php in the PHP sample and index.jsp in JSP sample).

### Example

```
<script src="viewer-assets/js/js/jquery.hotkeys.js"></script>
```

 Do not obtain the file from CDN. 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.

As an example, let us walk through 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;  
});
```

We can also change the code above to trigger the event on the whole document object.

### Example

```
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 we do not want the page navigation to occur when the modal dialogs are showing, in line 4, we check for the visibility of the modal dialogs.

Because our elements are divs and are not normally focusable, we 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. But be mindful, 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 'keyup' event. We are not using the 'keypress' event 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 a little 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

It is possible to build a custom Viewer UI by directly embedding the **ViewerControl** instead of embedding the Viewer using the jQuery plugin.

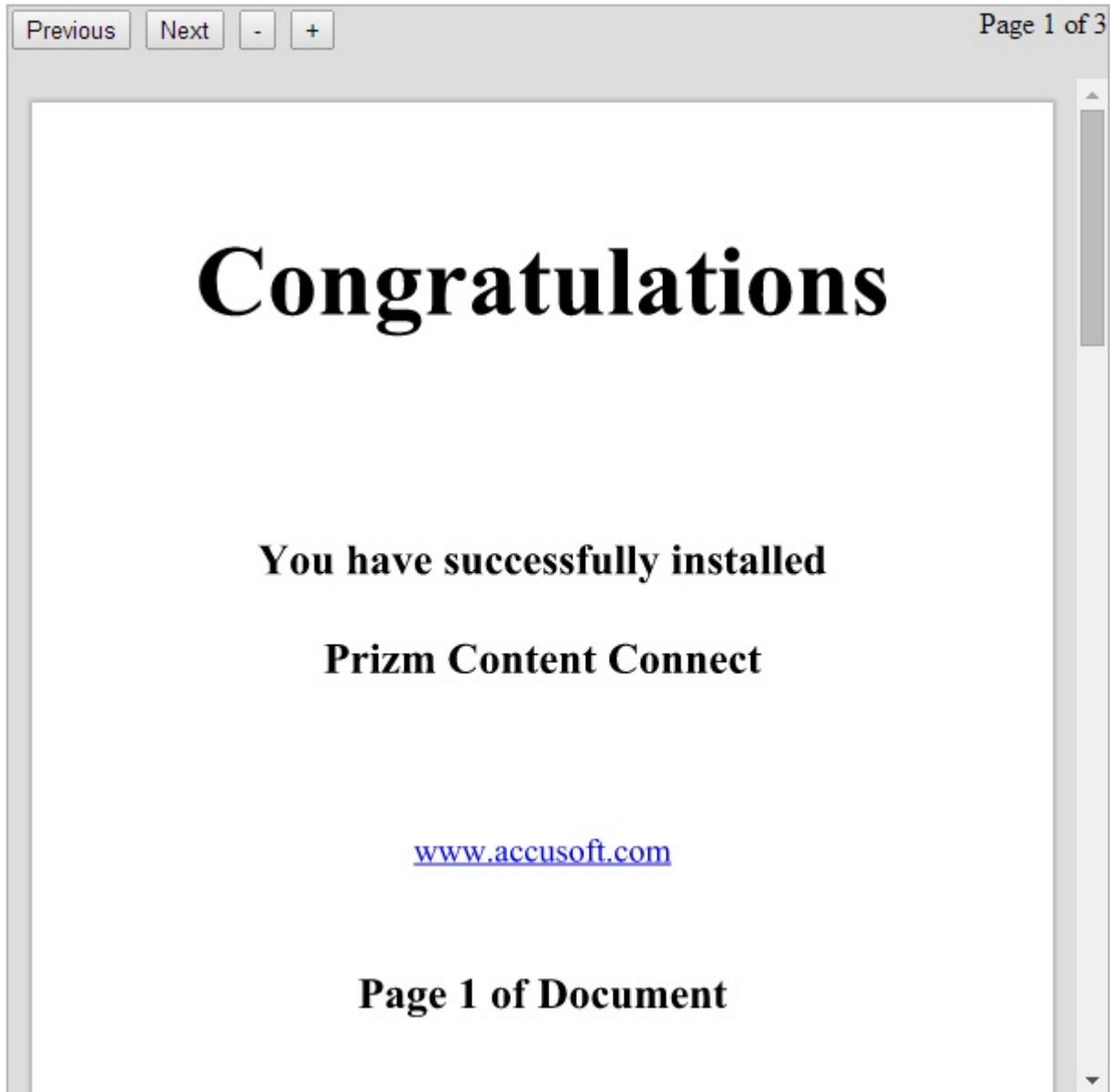
There are several benefits to building a custom Viewer UI, including the ability to:

- 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).

frameworks.

## 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**)
    - simple.js (**custom - shown below**)

 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>
  <!--Use jQuery to build the UI of the viewer-->
  <script
src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/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: "pcc.ashx",
    // printTemplate : "...", // include a print template for
```

```
    },
    // 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
        });
    });
});
```

 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;
```

```
float: right;
}
.floatRight {
  float: right
}
```

## 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>
```

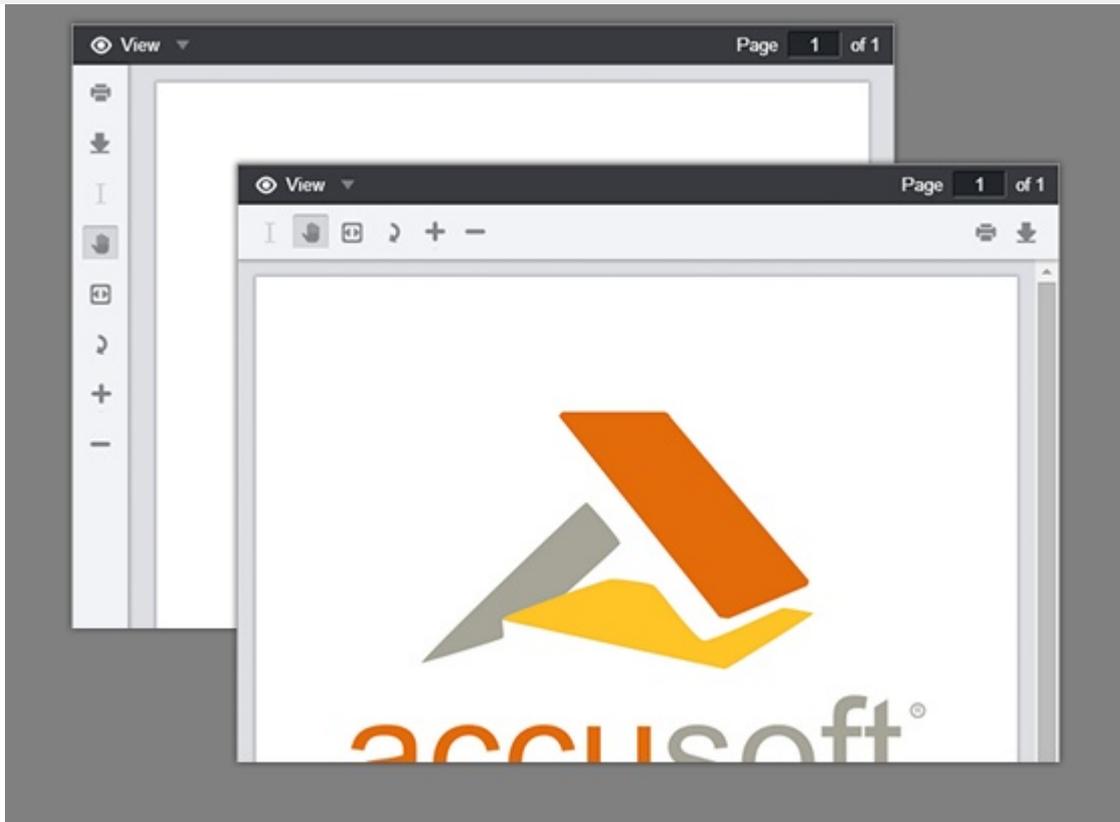
### Default Data Attributes

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.

## Change the Position of the Menu Bar

You can change the layout of the tab panes by modifying the markup located in **viewerTemplate.html**.



- Adding the **pcc-tab-vertical** class makes the tab pane vertical.
- Adding either a **pcc-left** or **pcc-right** class specifies 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

A custom mouse tool can be used in the Viewer, first by defining the tool and then by updating the UI to have a button for the tool.

**Step 1: Define the tool in JavaScript code.**

## 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 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

You can add a new custom tab to the Viewer tab navigation by modifying the markup located in **viewerTemplate.html**. 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">
```

```
    </div>
    <div class="pcc-tab-pane">
      <!-- Tab content -->
    </div>
  </div>
  <!-- End custom tab -->
</div>
```

## Customize the Markup

All of the markup in the Viewer is 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

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. 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 at <http://underscorejs.org/#template>.

#### Example

```
<!-- An example of a variable -->
<button data-pcc-rotate class="pcc-icon pcc-icon-rotate"
  title="<%= rotate %>"></button>
```

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>
```

Using the above example, 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**, 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"
```

```
class="foo" data-bbox="145 117 523 144" style="border: 1px solid #ccc; padding: 5px;">title="My Line Tool"></button>
```

## Customize the Styles

### Customizing the Styles

#### Working with the LESS preprocessor

The Viewer uses **LESS** to pre-process the CSS for the Viewer. In order to facilitate using this pre-processor in a development environment, the following files are included in the viewer-assets folder:

- Gruntfile.js
  - package.json
1. In order to use these files, you will need to install **Node.JS** in your development environment. Then, you can run the following commands from a command line or terminal:

#### Example

```
npm install -g grunt-cli  
npm install
```



Point the command line interface to the viewer-assets folder in order to execute these commands.

2. Next, you can use this command to build the CSS files required for production:

#### Example

```
grunt buildprod
```

3. You can also build development files, which will include extra source maps helpful in debugging CSS:

#### Example

```
grunt builddev
```

4. 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 files:

- css/viewer.css - contains the bulk of the Viewer styles
- css/fonts.css - contains fonts necessary for printing

## Customizing the Styles

The styles should be loaded in the Viewer in the following order:

1. normalize.min.css
2. viewercontrol.css
3. viewer.css
4. legacy.css (in IE8 only)

### Namespace

The Viewer uses the class **.pccv** in order 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 build 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** - The files contain the large Viewer components, and are named in a self-explanatory way. For example, styles related to the search functionality are help 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\*.png** files. The icons sprite image has a dark version for use on light backgrounds and a white version for use on dark backgrounds. There is a larger @2x version, to account for HD/Retina displays, and a regular sized version for legacy support. Since modern browsers support the background-size property, we use the @2x images for all icons and degrade to the regular sized icons for Internet Explorer 8.

### 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

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 IE8 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 Internet Explorer 8 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.

## viewercontrol.css

This file contains the styles required for using ViewerControl directly. This file should not be changed directly, but rather, should have any necessary rules overridden by your own CSS. If choosing not to use our Viewer, and instead embedding ViewerControl directly in a custom integration, this CSS file is still required.

## Polyfills

There are a few polyfills used to provide support for modern browser features:

consistency in the default styling of HTML elements.

## Disable the Print Button

### Introduction

There are two ways of disabling printing inside the Viewer:

- The first would be to remove the print button directly from a viewer template file.
- The second option would be to pass it in as a configuration parameter to the jQuery plugin.

Let's have a closer look at how both of these options can be used.

### Modifying the Printing Template

In order to remove printing from a template, you will need to open the "**viewerTemplate.html**" file that can be found inside a Samples folder. The exact location will depend on the type of sample you are using. For example, the .NET/Webforms sample it will be located at:

**...Prizm\Samples\dotnet\webforms\full-viewer-sample\viewer-assets\templates**

Once you've opened a file, search for an element with the attribute **data-pcc-print="launch"**. In the default template, it's located on line 10. Delete this element and restart the Viewer.

### Configuring Viewer Parameters

In order to disable printing via the Viewer parameters **options object**, that is passed to the Viewer, add the following **uiElements** parameter to the options object:

#### Example

```
var pluginOptions = {
  uiElements: {
    printing: false
  }
};
```

For example, in '**dotnet/webforms/full-viewer-sample**' samples, this options object is being created in '**Default.aspx**' file on line 132.

#### 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.

## Scroll the Viewer Programmatically

- 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.

### Example

```
// Scroll down by 100 pixels  
myViewerControl.scrollTo(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.

## Work with Annotations Programmatically

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.

### Creating and Updating Annotations

All annotation types, except `PCCViewer.Mark.Type.HighlightAnnotation`, can be created using the method **PCCViewer.addMark**. 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). Currently, the **HighlightAnnotation** can only be created programmatically after a search and this type of annotation can extend to more than one page.

### 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  
- 50});  
            //provide the label for the stamp mark  
            stampMark1.setLabel("Reviewed");
```

```
stampMark1.setBackgroundColor( "#110000" );
// ... 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 fillcolor 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);
}
);
```

## Programmatically Creating Highlight Annotations

The following example shows how to programmatically create highlight annotations:

### Example

```
var requestObject1 = viewer1.search("happy");
requestObject1.on (PCCViewer.EventType.SearchCompleted, function (event) {
var searchResults = event.completedSearchResults;
//create highlight annotation from the 2nd search result
var highlightMark = viewer.convertToHighlight(searchResults[1]);
}
);
```

## Reading 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();
```

## Selecting Annotations

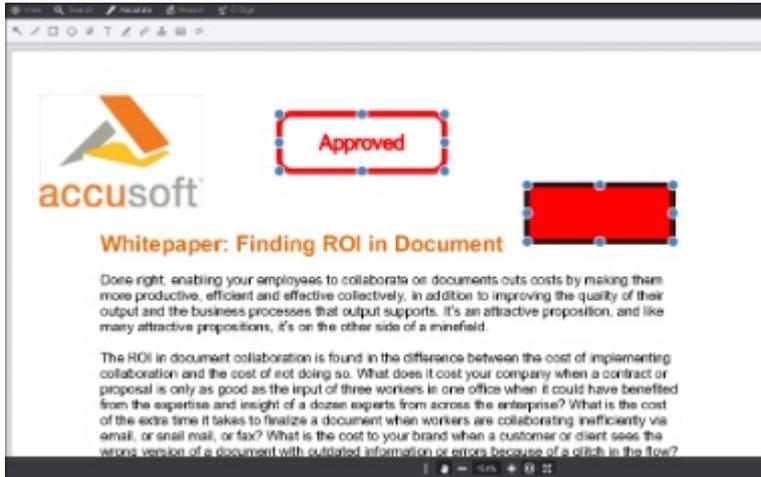
The following example shows how to select annotations:

### Example

```
//get all marks
```

```
//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:



## Deselecting Annotations

The following example shows how to deselect annotations:

### Example

```
//Get theselected 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");
    }
}
```

## Reordering 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
```

```
//use moveMarkBackward to move the mark one slot backward
viewer1.moveMarkBackward(rectangleMark1);
```

## Deleting Annotations

The following example shows how to delete annotations:

### Example

```
//In this example we will delete all the marks that are selected
//Get theselected marks
var selectedMarks = viewer1.getSelectedMarks();
//delete if there were any selected
if(selectedMarks.length > 0 ) {
    //delete all the selected marks
    viewer1.deleteMarks(selectedMarks);
}
```

## Saving Annotations

You can save annotations created in the currently displayed document using the **saveMarkup** method:

### Example

```
var savedRecordName = 'my annotations';
returnValue = viewer1.saveMarkup(savedRecordName);
returnValue.then(
    function(recordName) {
        //succeeded
        savedRecordName = recordName;
    },
    function(rejectedReason) {
        //failed
        alert("There was error in saving annotations, error: " +
rejectedReason);
    }
);
```

The markup is saved to an XML file using a filename that includes a unique ID for the document being viewed and the provided annotation record name, so valid filename characters are required. When later viewing the document, the saved markup can be loaded by passing the record name to the `PCCViewer.ViewerControl#loadMarkup` method.

## Loading Annotations

Any previously saved annotations record associated with the document can be used to load annotations using the **loadMarkup** method:

### Example

```
var currentlyLoadedAnnotations;
```

```
function(recordName) {
    //succeeded
    currentlyLoadedAnnotations = recordName;
    //update the UI now if you wish
},
function(rejectedReason) {
    //failed
    alert("There was error in loading annotations, error: " +
rejectedReason);
}
);
```

## 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

```
<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.

### 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);
```

```
viewer.on(PCCViewer.EventType.ViewerReady, function() {
    // 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.

## 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

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
"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

## Modify viewer.js

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.

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](#)
- [Initialization and Binding the Markup](#)
- [Auxiliary Functions](#)
- [ViewerControl Event Handlers](#)
- [Create the ViewerControl and Add Listeners](#)
- [Search and Annotation IO Modules](#)
- [jQuery Plugin](#)

### 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

**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](#).

## Annotation Layers

PrizmDoc's Annotation Layering functionality makes it possible to create, view and manage multiple sets of annotations for a document enabling a new world of 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 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.

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).

## New Annotation File Format

When adding the Annotation Layering functionality, we revamped our annotation storage mechanism to accommodate cross-layer references for commenting. For this reason, we have moved to a new JSON file format for all annotation persistence. This JSON format is based on our new 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 now use the new JSON file format. This means that when an XML file is loaded for editing, saving it again will save it as a new 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:

- [Working with Annotation Layers](#)
- [Loading Annotations from the Web Tier](#)
- [API - jQuery.fn](#)

## 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(
```

```
// Load a specific record so that the data is available to the HTML
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);
}
);
```

## PrizmDoc Application Services

Note that PrizmDoc Application Services is installed "ready to run" via any of our 4 web tier samples (C#, MVC, Java, PHP). This section describes how you can set up your server to handle the proxying these samples implement and eliminate the need for our web tier samples altogether.

This section contains the following "How To" information:

- [Set up Your Database for use with PrizmDoc Application Services](#)
- [Migrate to PrizmDoc Application Services](#)
- [Handle Specific Routes from PrizmDoc Application Services using Custom Logic](#)
- [Pre-Convert Documents](#)
- [Viewing Packages](#)

For information on developing against PrizmDoc Application Services, refer to [Administering PrizmDoc > PrizmDoc Application Services](#).

## Set up Your Database for use with PrizmDoc Application Services

### How to set up your database for use with PrizmDoc Application Services

For information on configuring PAS to communicate with your database, see the [PAS Configuration](#) section in the help file.

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
```

## 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
```

```
/usr/share/prizm/pas/db/mssql-scripts
```

## Migrate to PrizmDoc Application Services

For existing customers of PrizmDoc who have already done the work to integrate the Viewer into their applications, there is naturally a question about how to, and perhaps more importantly, why migrate to the new PrizmDoc Application Services architecture.

The first thing to know about PrizmDoc Application Services is that there is no requirement that your application migrate to use it. PrizmDoc Application Services makes use of the same RESTful API that your application currently supports, and that won't change in the future (although we will likely augment this API over time to include more functionality).

However, there are several strong reasons why it's beneficial to make the switch and use PrizmDoc Application Services:

- You will know that your application always has the canonical approach to working with the Viewer. No wondering about any odd nuances in the REST APIs that you might be missing.
- PrizmDoc Application Services will always incorporate any changes to the APIs used by the PrizmDoc Server conversion services, relieving your team of having to track those changes for most document viewing tasks.
- Migrating to PrizmDoc Application Services will actually remove significant code from your application, which will reduce your future maintenance costs.
- And perhaps most importantly: we have several ideas on how to utilize PrizmDoc Application Services to add significant functionality to the overall PrizmDoc experience, some of which we're working on already. Without incorporating PrizmDoc Application Services in your application, you will not be able to take advantage of these new features.

### How to Migrate to PrizmDoc Application Services

The work needed to migrate your application to PrizmDoc Application Services can range from simply overriding several routes in your web server's configuration to having to override several functions from the PrizmDoc Application Services implementation and redirecting them to your own. See which one of the following scenarios best explains your situation:

#### Scenario 1

If your application has used the web sample code we've provided in the past without modifications, then the migration path is simple: install PrizmDoc Application Services and see the help article on [how to configure it for your particular web server](#).

#### Scenario 2

If you are using a custom storage mechanism for your documents, but not markup files, you might be in a situation where you are using a custom markup id when creating a session. In the previous samples, this is the value of `origin.documentMarkupId`. If this is the case, you can update your code for creating a

the PrizmDoc Server. You can provide the same value to `source.markupId`, allowing PrizmDoc Application Services to use the same custom ids you have used in the past. For more information, see the PrizmDoc Application Services [Viewing Session API](#).

### Scenario 3

Things begin to get more complicated if you've implemented your own storage mechanism for annotations and/or image stamp files. While PrizmDoc Application Services will be backward compatible with these files when created by our web samples, there's no way for it to know how you've stored your own documents. In this case the only recourse is to [override the appropriate calls into PrizmDoc Application Services with calls to your own code to handle persistence](#).

Note that even in this scenario you can take advantage of most of PrizmDoc Application Services functionality, as you'll only be overriding a few functions at most.

## Handle Specific Routes from PrizmDoc Application Services using Custom Logic

### How to handle specific routes from PrizmDoc Application Services using custom logic

PrizmDoc Application Services is designed to handle all viewer requests in an appropriate way. However, there might be some cases where you need to be able 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 PrizmDoc Application Services

In the API References, 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 PrizmDoc Application Services 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, this file would be `C:\Prizm\pas\pcc.win.yml`, and on Linux, this would be `/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 PrizmDoc Application Services to use it. This can be configured as such:

```
# DEFINE ANY HEADER YOU WOULD LIKE
```

```
routeHandlers.RouteKey.headers.x-my-custom-secret: "c2hoLCBJJ20gYSBzZWNYZXQ="
routeHandlers.RouteKey.headers.x-use-any-name: "YW55IG5hbWUsIHJlYWxseQ=="
```

PrizmDoc Application Services will proxy any request to `RouteKey` to the url specified in the config. Since PrizmDoc Application Services routes will sometimes include important information -- such as document ids, markup ids, etc. -- the entire route from PrizmDoc Application Services will be appended to the url, as such:

```
// route as handled by PAS
http://localhost:3000/RouteKey/u1234/s0mEiDvAlUe

// it will be forwarded here, as per the above config
http://myserver/some/route/RouteKey/u1234/s0mEiDvAlUe
```

If you would prefer to defer handling of that request back to PrizmDoc Application Services, you can return a `202 Accepted` response to instruct PrizmDoc 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 that 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 PrizmDoc Application Services are open to all users, since the PrizmDoc Application Services service does not know about system users. However, you can let PrizmDoc Application Services 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 PrizmDoc Application Services on how to continue:

```
var markupType = parsedRouteParameters.markupType; // "FormDefinitions" or
"MarkupLayers"
```

```
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 PrizmDoc Application Services access to your documents directly -- for example, when documents are in a custom content management system -- and 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);
```

## Pre-Convert Documents

### Pre-converting documents in Application Services

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 viewingPackages:

1. How to Create a viewingPackage by Pre-converting Documents.
2. How to Obtain Information for a viewingPackage.
3. How to Delete a viewingPackage

### How to Create a viewingPackage 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
```

```
Content-Type: application/json
{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.doc",
      "documentId": "unT67FxeKm81k1p0kPnyg8",
      . . .
    },
    "viewingPackageLifetime": 2592000
  }
}
```

A successful response to the above POST provides '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.

```
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.

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 viewingPackage

For obtaining detailed information about the converted viewingPackage, 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
```

### Response Body

```
200 OK
Content-Type: application/json
{
  "input": {
    "source": {
      "type": "document",
```

```
    "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 viewingPackage, 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)
```

## Viewing Packages

A Viewing Package is a cached version of a document that the PrizmDoc 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 on Multiple Servers](#) topic can provide more information for configuring PAS in Multi-Server Mode.

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"
```

```
viewingSessionsData.storage: "database"
viewingSessionsProcessesMetadata.storage: "database"

viewingPackagesArtifactsMetadata.storage: "database"
viewingPackagesArtifacts.storage: "filesystem"
viewingPackagesArtifacts.path: "/usr/share/prizm/Samples/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 PrizmDoc Application Services 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.

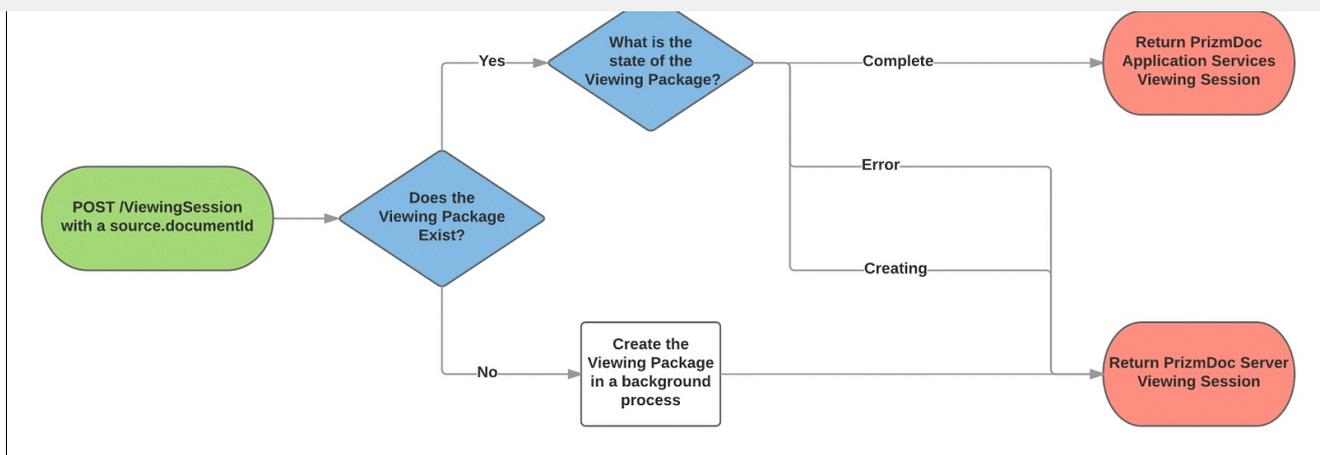
## On-Demand Caching

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-a1b0x19n2",
    "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, PrizmDoc Application Services 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

### 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.

## PrizmDoc Server

This section contains the following information:

- [Use the PrizmDoc Server API](#)
- [Markup Burner XML Specification](#)
- [How To](#)
  - [Convert Content with Content Conversion Service](#)

- [How to Configure the Demo on Windows](#)
- [How to Configure the Demo on Linux](#)
- [Migrate from Accusoft Cloud-Hosted Servers to Self-Hosted Servers](#)
- [Natively Render Microsoft Office Documents](#)
- [Perform Auto-Redaction](#)
- [Pre-Populate Fields in the E-Signature Viewer](#)
- [Set up a Viewing Session for a CAD Drawing which has XREF Dependencies](#)
- [Transfer Your Document to PrizmDoc Server](#)
- [Use a Viewing Session](#)
- [Watermark Content in a Viewing Session](#)

## Use the PrizmDoc Server API

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](#)
- [PrizmDoc Server Operations](#)

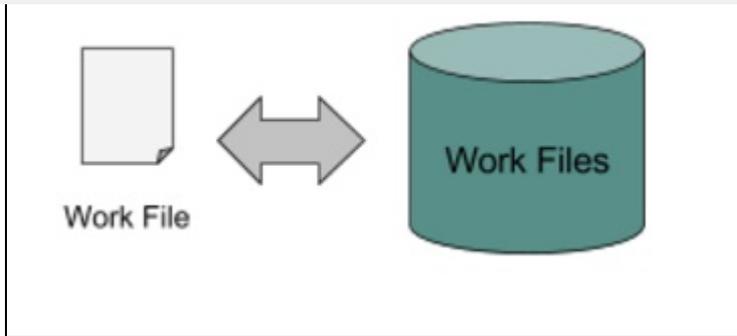
### Resources

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](#)
- [Processes](#)
- [Viewing Sessions](#)

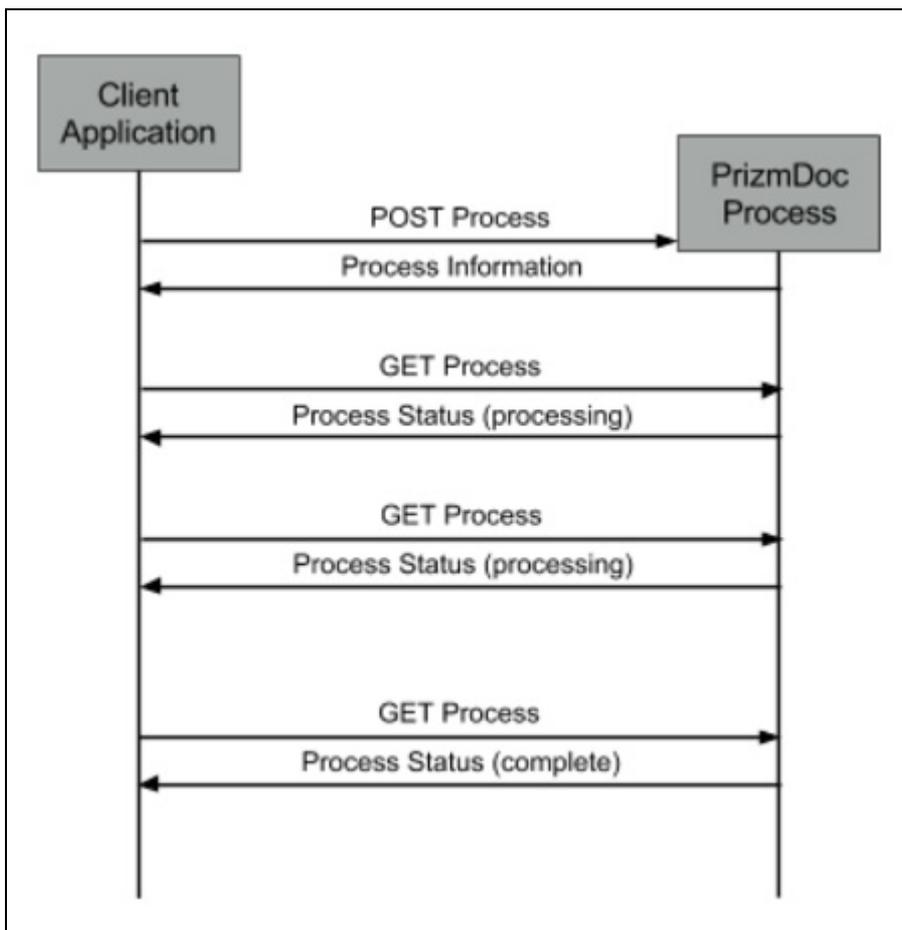
### Work Files

[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 operations.



### Processes

Several PrizmDoc 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.



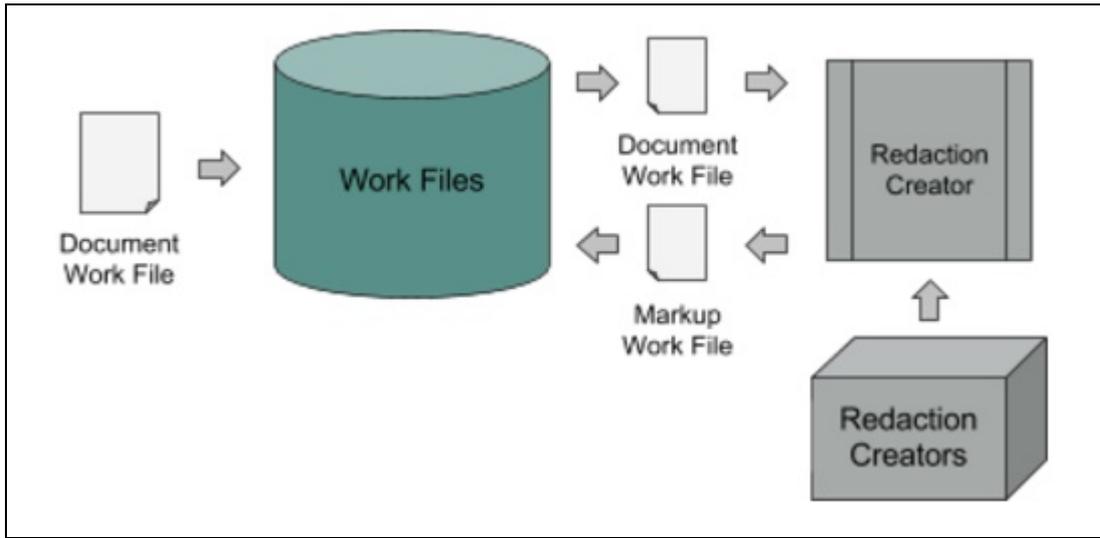
These processes include:

- [Redaction Creators](#)

- Content Converters

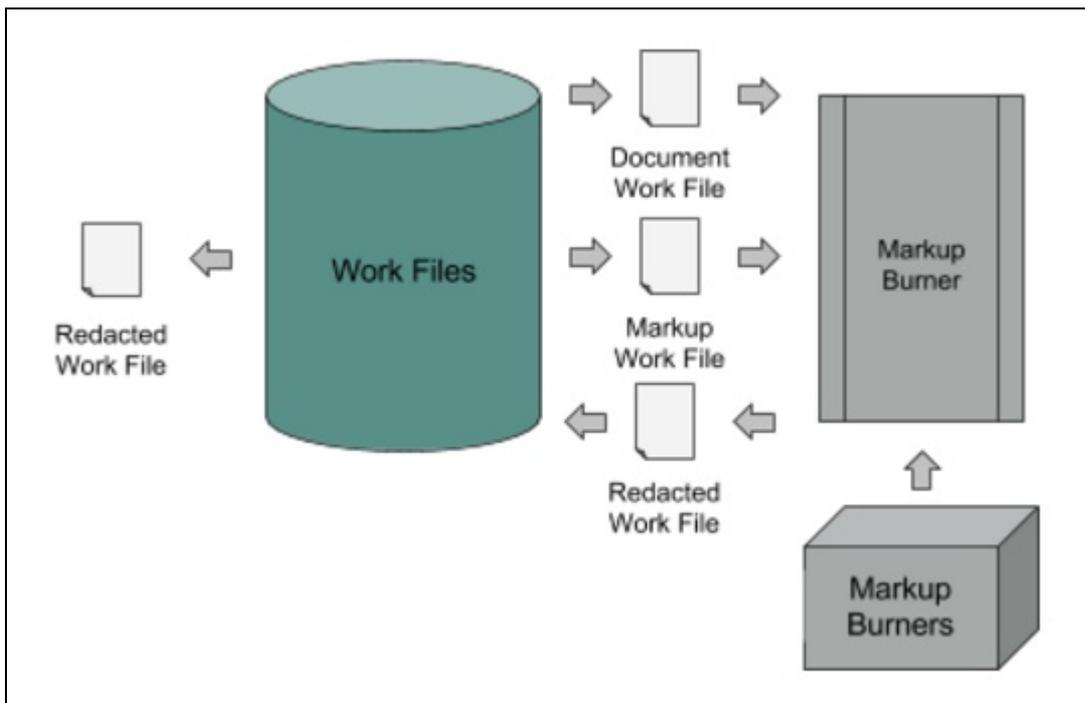
## Redaction Creators

**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.

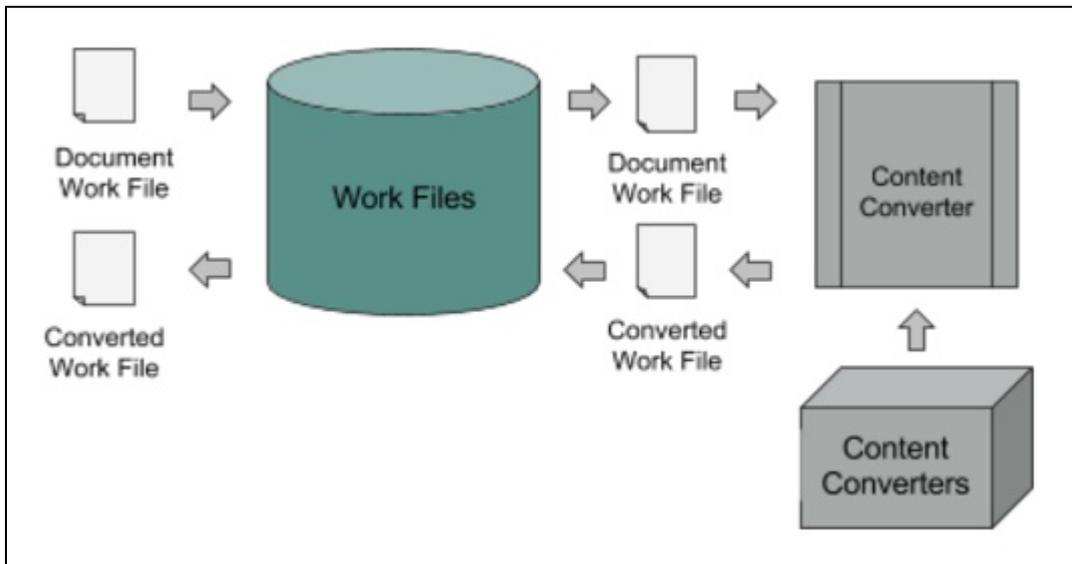


## Markup Burners

**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.

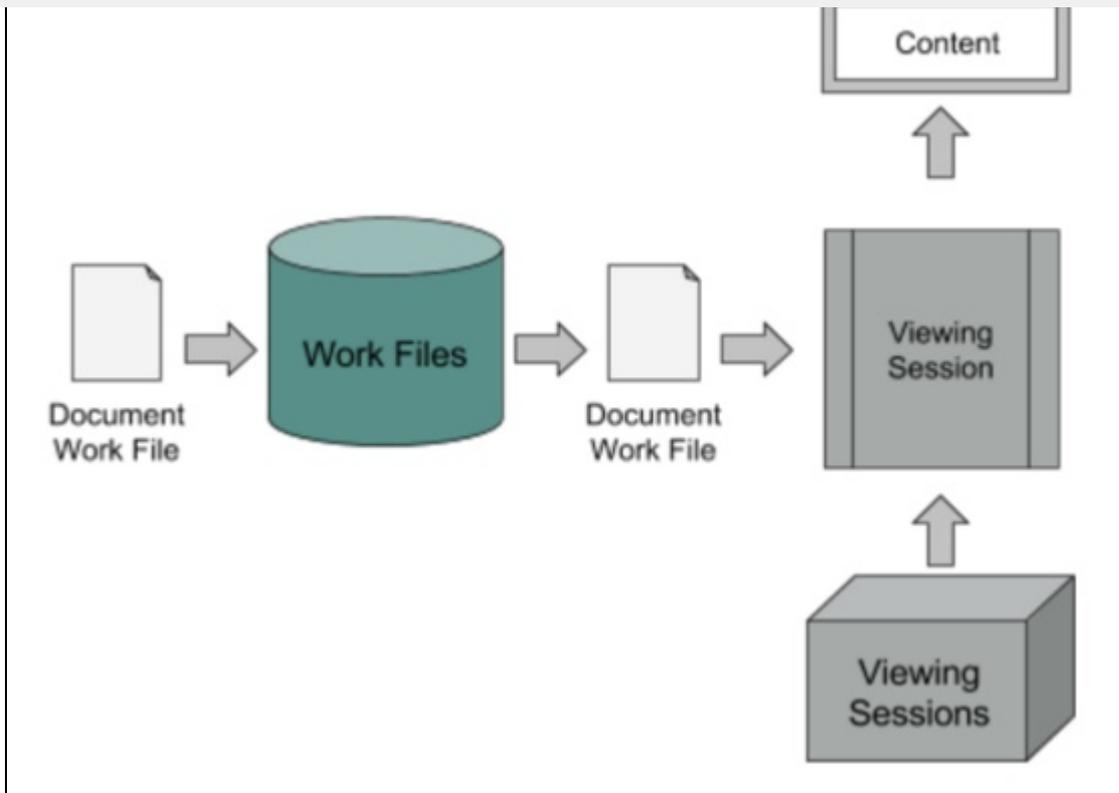


**Content Converters** are processes that convert one or more source Work File documents to one or more output Work File documents. Many different document formats are supported as source documents, and several common formats are allowed for output. After process completion, converted output is available 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 RESTful 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 instances in a **multi-server environment**, APIs exist to query and configure the servers that each instance can use to route traffic for load balancing.

## Markup Burner XML Specification

### Markup Burner XML Specification

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

Name	Type	Required	Description
id	integer	yes	A 1-based page number; annotations that are listed as child elements of a given <page> are only applied to the corresponding page of the target document.
pageWidth	floating point	yes	Page width. This attribute is only used for positioning of annotations which use absolute coordinates (such as <line-rectangles> data in <a href="#">text_hyperlink_annotation</a> ). This value can be arbitrary, as long as absolute coordinates specified in annotations are consistent with it.
pageHeight	floating point	yes	Page height. This attribute is only used for positioning of annotations which use absolute coordinates (such as <line-rectangles> data in <a href="#">text_hyperlink_annotation</a> ). 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
drawType	string	yes	One of the supported draw types (see <a href="#">Draw Type Descriptions</a> below) that defines what modification should an annotation apply to a page.
x_percent	float	0.0	X position of a bounding rectangle, in 0-to-1 based portion of a page width.
y_percent	float	0.0	Y position of a rectangle, in 0-to-1 based portion of a page height.
width_percent	float	0.0	Width of a bounding rectangle, in 0-to-1 based portion of a page width.
height_percent	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:

```
<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>
```

## 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
imageStampId	string	ID of an image that will be used by annotations to reference to it.
base64format	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".

### arrow

Draws a **line** with a triangle-shaped head. Same set of attributes, the size of an arrowhead is hardcoded 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;

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
eSignWidth	int	280	Overrides the boundary rectangle width
eSignHeight	int	10	Overrides the boundary rectangle height
eSignIdLabel	string	eSign ID:	Signature label (first part of the signature annotation text)
eSignUuid	string	<empty string>	Signature uuid (second part of the signature annotation text)
eSignDate	string	<current date>	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 <a href="#">text_redact</a>
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">
```

```
</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 scaling factor when scaling the path to fit into the boundary rectangle.
lineWidth	int	1	Width of the path line, in pixels
lineColor	color	black	Color of the path line
align	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"> <li>• right</li> <li>• left</li> </ul> <p>Missing the parameter or setting it to any other value will result to default behavior (center aligned path line).</p>

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:

imageStampId	string	Yes	id of an image to use (see <a href="#">Image Binary Data</a> 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
lineWidth	int	1	width of the line, in pixels
lineColor	color	black	color of the line
dragDirection	string	No	Defines the direction of the line. Defines the placement of the arrowhead when used for an <a href="#">arrow</a>

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>
```

Note:

- Polyline coordinates are specified in points
- Boundary rectangle attributes are ignored for this annotation.

### Supported attributes:

lineColor	color	black	color of the line
lineWidth	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
lineColor	color	0	rectangle border color
fillColor	color	0	rectangle fill color
lineWidth	integer	0	Width of border in pixels
alpha	integer	255	Alpha component that is added to fill and line colors. The default value is 255 for 100% 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 pages' 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	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

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
lineWidth	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	Default	Description
href	yes	string	HTTP URL
lineColor	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.
fillColor	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, not supported by Redaction API at the moment.

## 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) **is set in a html-like piece of markup** in a CDATA child element of the `<annotation>`, that goes 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	/ Default	DESCRIPTION
ALIGN	string	No	<p>Controls the horizontal alignment of a text entry within the bounding rectangle area.</p> <p>Can have one of the following self-explanatory values:</p> <ul style="list-style-type: none"> <li>• RIGHT</li> <li>• MIDDLE or CENTER (synonyms)</li> </ul> <p>Setting ALIGN to any other value will result to default behavior (left aligned text).</p>
FACE	string	Arial	<p>Name of the font to use. For purposes of this annotation, the following font names can be used:</p> <ul style="list-style-type: none"> <li>• a font that is available to the OS;</li> <li>• a font that resides in a OS-specific fonts folder;</li> <li>• a font that resides in a specified fonts folder of a product (usually ./modules/fonts).</li> </ul>
SIZE	float	20	Font size to use
COLOR	string	#000000	<p>a usual HTML hex representation of an RGB color of a font to use. Note:</p> <ul style="list-style-type: none"> <li>• this attribute uses a HTML-like hex color notation unlike the -color attributes of other annotations;</li> <li>• HTML color names are NOT supported;</li> <li>• 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>.</li> </ul>

## text\_selection\_redaction (reserved for internal use)

Reserved for internal use, not supported by Redaction API at the moment.

## How To

This section contains the following "How To" information:

- [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 Accusoft Cloud-Hosted Servers to Self-Hosted Servers](#)
- [Natively Render Microsoft Office Documents](#)
- [Perform Auto-Redaction](#)
- [Pre-Populate Fields in the E-Signature Viewer](#)
- [Set up a Viewing Session for a CAD Drawing which has XREF Dependencies](#)

- [Use a Viewing Session](#)
- [Watermark Content in a Viewing Session](#)

## Convert Content with Content Conversion Service

This section provides steps for converting content with Content Conversion Service.

### 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.

```
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.

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 ContentCoverter 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"  
    }  
  ]  
}
```

```
    case : {
      "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"
    }
  }
}
```

### 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": "qRqGmJk0CABZ0001_0Aq",  
  "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:

```
"input": {  
  "sources": [  
    {  
      "fileId": "EYsfBhL0JbYgNk80sbnxEg"  
    }  
  ],  
  "dest": {
```

```
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:

```
"input": {  
  "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:

```
"input": {  
  "sources": [  
    {  
      "fileId": "EYsfBhL0JbYgNk80sbnxEg"  
    }  
  ],  
  "dest": {  
    "format": "pdf",
```

```
      "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:

```
"input": {
  "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"
      }
    ]
  }
}
```

```
ExpirationDate: 2017-05-24T15:22:02.002Z ,
"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"
    }
  }
}
```

## Content Conversion Demo

This section contains the following information:

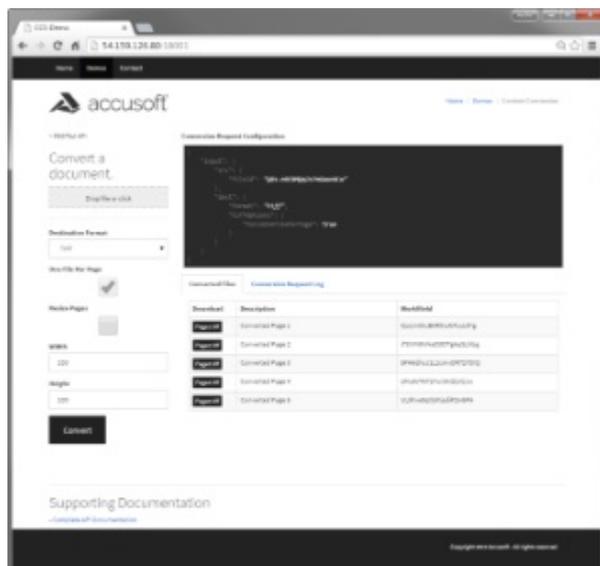
- [How to Configure the Demo on Windows](#)
- [How to Configure the Demo on Linux](#)

## 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** on the **Start Menu**
- The following will route you directly to the CCS Sample test page: <http://localhost:18001>
- Upon the initial load of the CCS Demo with the above link, you will see this screen:



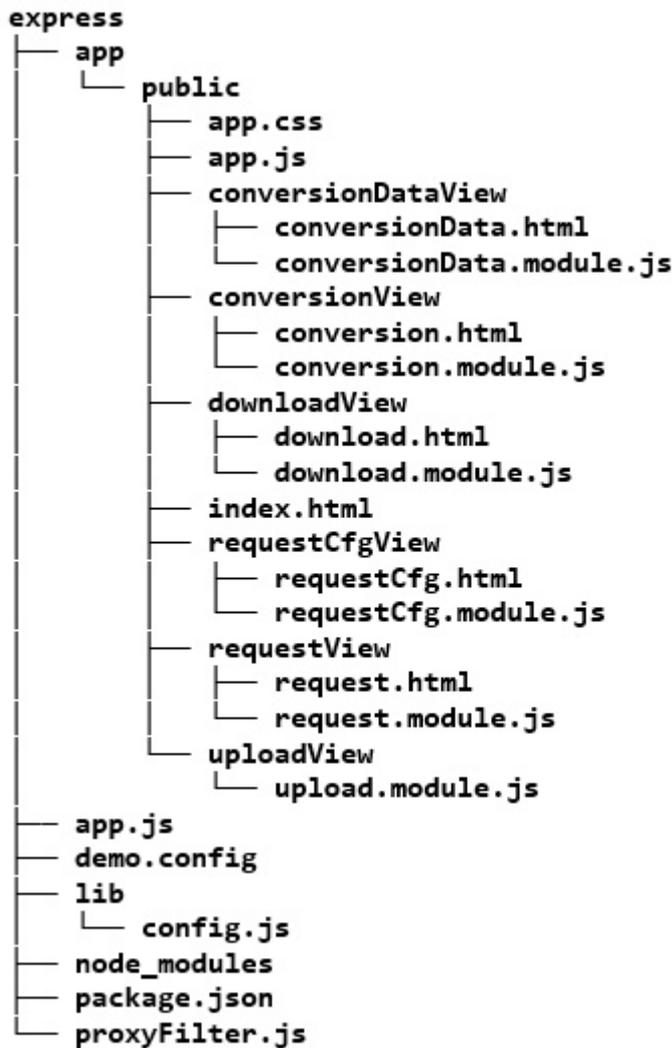
The demo.config file can be updated to customize the application host address, application port, and services port.

#### Example

```
{
  "logging": {
    "consoleLogFilePath":
"%ALLUSERSPROFILE%\Accusoft\Prizm\Logs\CssDemoService.console.log"
  },
  "httpService": {
    "port": 3000
  }
}
```

```
    "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.

	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 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

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:

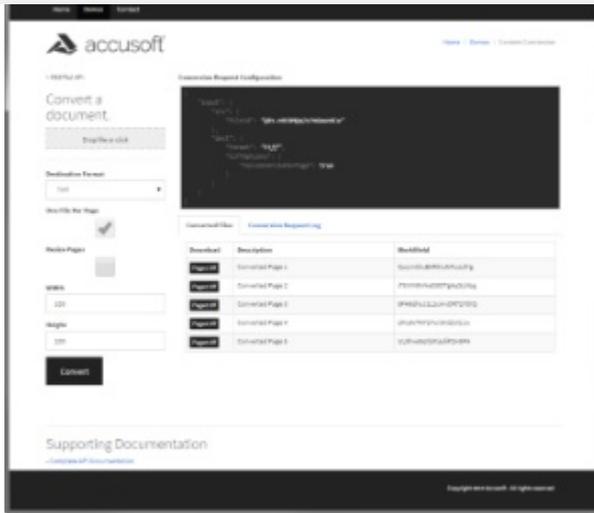
1. Log onto the system using an account with administrator privileges.
2. Click **Start > Run**.
3. Type **eventvwr**.
4. Press **Enter**.
5. Expand Applications and Services Logs.
6. Click **PrizmDemo**.

## How to Configure the Demo on Linux

### Demo Configuration

After the installation, test the sample application in a browser:

- The following will route you directly to the CCS Sample test page: <http://localhost:18001>
- Upon the initial load of the CCS Demo with the above link, you will see this screen:

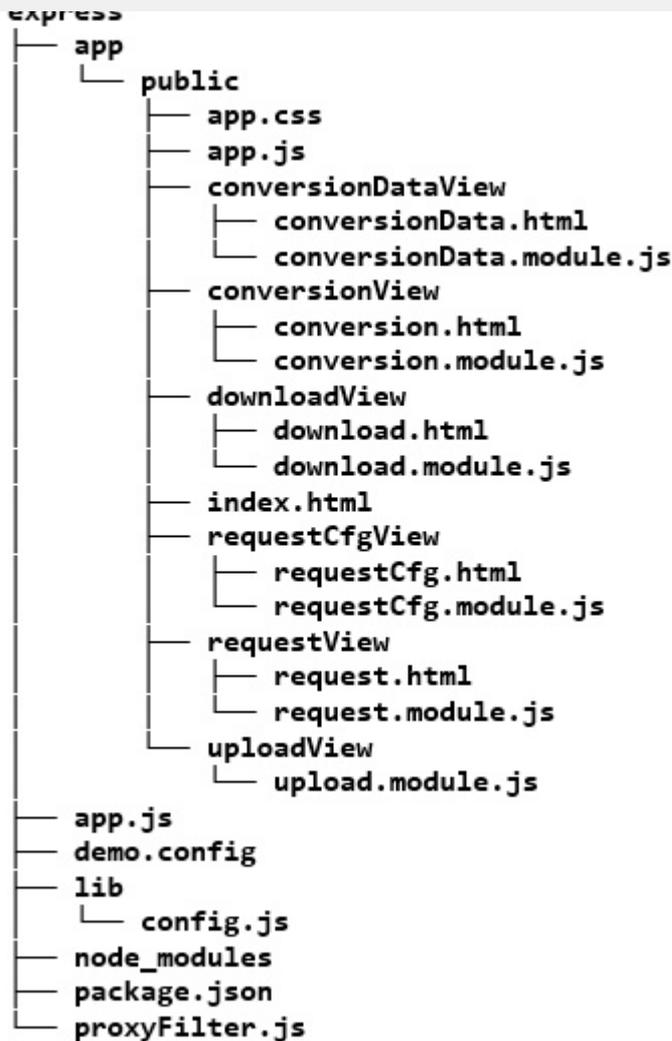


The demo.config file can be updated to customize the application host address, application port, and services port.

## 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/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

The Express Sample is automatically started as part of the setup.sh script found in the installation directory, typically **/usr/share/prizm/**. However after rebooting the Express sample will not be automatically restarted. To manually start and stop the Express sample, use the script found in **/usr/share/prizm/scripts/demos.sh**.

Starting the Express Sample from the shell:

### Example

```
$ /usr/share/prizm/scripts/demos.sh start
```

Stopping the Express Sample from the shell:

### Example

```
$ /usr/share/prizm/scripts/demos.sh stop
```

## to Self-Hosted Servers

### Step 1: Install PrizmDoc Server

You must install PrizmDoc Server first if you do not have it installed already. To install PrizmDoc Server on your server, refer to the section, [Installing PrizmDoc Server](#).

 Your Accusoft Cloud-Hosted key cannot be used as your PrizmDoc Server license, but you will have an option to install using an evaluation license. If you wish to obtain a PrizmDoc Server license after you try it out, please visit our [pricing guide](#).

### Step 2: Configure PAS to use your new PrizmDoc Server

Now that you have PrizmDoc Server installed, you just have to point your PrizmDoc Application Services (PAS) instance to your new PrizmDoc server. Use the following steps to accomplish this:

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. For instructions, refer to the topic, [Starting & Stopping PrizmDoc Application Services](#).

You are done! Your PrizmDoc Application Services are now pointing to your new instance of PrizmDoc Server.

## Natively Render Microsoft Office Documents

With PrizmDoc v12.0, Accusoft is offering a new 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 the license key that includes the MSO feature. The MSO feature can be purchased additionally to your standard PrizmDoc Server license.

The MSO conversion option 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

determine the required registry configuration settings specific to your server characteristics, make those changes in the registry, and require system reboot.

## Auto Deployment

In order to avoid a system reboot when installing PrizmDoc Server for Windows: for cloud-based systems using auto-deployment, all the required components listed in the [Windows Requirements](#) section should be installed and manually configured prior to creating an image of the Server system.

Additionally, the following registry modification is required to be made prior to creating an image of the Windows Server system to provide the recommended heap size scalability for the Microsoft Office rendering engine and achieve the best conversion performance results:

**Registry Key:** HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\SubSystems

**Value:** Windows

That 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 is required to be manually set to the Heap Size Value corresponding to the server specification as described in the following table:

CPU Cores	Instance Count	Heap Size Multiplication Factor	Heap Size Value
Cores < 4	Number of Cores	1	768 (default)
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 the heap size multiplier corresponding to the number of CPU cores available on the server.

 The manual heap size value adjustment process is not required when the PrizmDoc Windows Server installer is running in the interactive UI mode since the installer will make all the necessary registry changes and require the system reboot.

The instance count indicates the number of Office document converter processes the Office Conversion service will handle when using the MSO option.

## Perform Auto-Redaction

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.

The following steps will walk you through the auto-redaction process using the PrizmDoc Back-end RESTful Services.

### 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 Back-end RESTful Services, except for DICOM 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 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",
}
```

### 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/>)
- 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 that 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

```
"(?i:\bthe\b)"

JSON content:
"regex": "(?i:\\bthe\\b)";

C# code:
string regex = "(?i:\\\\bthe\\\\b)";
```

### Step 3: Create Markup XML 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 which requires two inputs:

- The **file ID** of source document you uploaded in Step 1, and
- The **regular expression** you created in Step 2.

#### Example

```
POST http://192.168.0.1:18681/PCCIS/V1/RedactionCreator
Content-Type: application/json
{
  "input": {
    "documentFileId": "5qTYa3gzN9gYUb5SzqUhqg",
    "autoRedactionRegularExpressions": ["[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"]
  }
}

200 OK
Content-Type: application/json
{
  "processId": "Rr64ma-U_HseoPrs6y0iiw",
  "expirationDateTime": "2014-12-03T18:30:49.460Z",
  "input": {
    "documentFileId": "5qTYa3gzN9gYUb5SzqUhqg",
    "autoRedactionRegularExpressions": ["[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"]
  },
  "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

- See the [Redaction Creator API](#) for more details of this request.

### Example

```
GET http://192.168.0.1:18681/PCCIS/V1/RedactionCreator/Rr64ma-U_HseoPrs6y0iiw

200 OK
Content-Type: application/json
{
  "processId": "Rr64ma-U_HseoPrs6y0iiw",
  "expirationDateTime": "2014-12-03T18:30:49.460Z",
  "input": {
    "documentFileId": "5qTYa3gzN9gYUb5SzqUhqg",
    "autoRedactionRegularExpressions": ["[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"]
  },
  "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. This is accomplished by sending a **POST request** which will start a process that runs asynchronously on the PrizmDoc server to produce a redacted document.

### Example

```
POST http://192.168.0.1:18681/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"
  },
}
```

```
    "percentComplete": 100,  
  }  
}
```

### 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 of this request.

#### Example

```
GET http://192.168.0.1:18681/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]
```

## How to Populate Fields in the PrizmDoc Signature

### Viewer

#### Example Integration (Runtime)

This example is to automatically pre-populate 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 client. 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) {
```

```
        CHECK (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]
                    }
                });
            // 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;
});
```

 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 client and the FormLoaded event in the E-Signature Viewer fires, fields in this form definition with a valid defaultValue will be populated.

## Set up a Viewing Session for a CAD Drawing which has XREF Dependencies

How to Setup a Viewing Session for a CAD Drawing Which Has XREF

This guide explains how to setup 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 guide 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 he 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 PCC 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
```

```
{
  "render": {
    "html5": {
      "alwaysUseRaster": false
    }
  }
}
```

```
HTTP/1.1 200
Content-Type: application/json
```

```
{
  "viewingSessionId": "AqQp3wrrSZ5YhJoFdj44Z_rT4629Xn06j7bEjjSRA5fiQUljuYgi-
ng9sP4v95VTVqilte6bsaC6eGpUu1MUWid1VN9qwH6rN5wp82eSfM",
  "affinityToken": "ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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 HTML5 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 Server Self-Hosted in multi-server mode or PrizmDoc Server Accusoft Cloud-Hosted, 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/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

```
<<master.dwg bytes>>
```

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
```

```
"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": "o1bLJwFGxf9QGUTkyr0qig",  
  "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/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

common/logo.png:

POST /PCCIS/V1/WorkFile

Content-Type: application/octet-stream

Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

<<common/logo.png bytes>>

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "fileId": "JcskB6w8D5_q1f40A3xspQ",
  "fileExtension": "png",
  "affinityToken": "ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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

ACCUSOFT-AFFINITY-TOKEN: "ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="

```
{
  "primaryPath": "master.dwg",
  "items": [
    { "fileId": "CVBuD7DbQYNoJDqByGierQ", "path": "master.dwg" },
    { "fileId": "5qTYa3gzN9gYub5SzqUhgq", "path": "parts/desk.dwg" },
    { "fileId": "o1bLJwFGxf9QGuTkyr0qig", "path": "parts/desk-extension.dwg" },
    { "fileId": "KxonBTYJyRpCsw0Ln_paiw", "path": "other/chair.dwg" },
    { "fileId": "JcskB6w8D5_q1f40A3xspQ", "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": "5qTYa3gzN9gYub5SzqUhgq", "path": "parts/desk.dwg" },
    { "fileId": "o1bLJwFGxf9QGuTkyr0qig", "path": "parts/desk-extension.dwg" },
    { "fileId": "KxonBTYJyRpCsw0Ln_paiw", "path": "other/chair.dwg" },
    { "fileId": "JcskB6w8D5_q1f40A3xspQ", "path": "common/logo.png" }
  ],
  "affinityToken": "ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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 carefully that the viewingSessionId used in the URL is prefixed with the letter u:**

```
PUT /PCCIS/V1/ViewingSession/uAqQp3wrrSZ5YhJoFdj44Z_rT4629Xn06j7bEjjSRA5fiQUljuiYgi-
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 creating a new viewing session. **If you are using PrizmDoc Server Self-Hosted in multi-server mode or PrizmDoc Server Accusoft Cloud-Hosted, 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/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

```
{
  "render": {
    "html5": {
      "alwaysUseRaster": false
    }
  }
}
```

```
HTTP/1.1 200
Content-Type: application/json
```

```
{
  "viewingSessionId": "H09MagQSyIizShhzLpjp-H73-
IRtoqcPlJZmLP0PPwI3HxJ36Ds_HunbqMmtKfT1gt4h4-96X67t7onW2P9XfV5Rw4pSddBrNoFp4A8bdFw",
  "affinityToken": "ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

### Step 2: Attach the Existing 'Package' Work File to the Viewing Session

viewingSessionId used in this URL must be prefixed with the letter u:

```
PUT /PCCIS/V1/ViewingSession/uH09MagQSyIizShhzLpjp-H73-IRtoqcP1JZmLP0PPwI3HxJ36Ds_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 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

prevent this, make sure you are providing the work item's `AFFINITY-TOKEN` value in an `ACCUSOFT-Affinity-Token` header when making the POST request to create the new viewing session (cf. Step 1).

## Transfer Your Document to PrizmDoc Server

PrizmDoc Server will expect a source document to be provided for each new viewing session using one of three available methods. The specific method of transfer used for a particular document is specified during the creation of the [viewing session](#), so it is possible to use multiple transfer methods in your application. You can transfer your document to PrizmDoc Server by sending it explicitly with a HTTP PUT request or by specifying a file location, such as an HTTP URL or a local filename, so that PrizmDoc Server can retrieve it.

The best transfer method for your application will likely depend on the original location of your documents. For example, if your source documents are stored in a database, the HTTP PUT request is likely the most efficient method as you'll be able to read the data out of the database and copy it directly to the request body, eliminating the need to write the document to a file. If your source documents are already hosted and accessible from an HTTP URL, PrizmDoc Server can download the file directly thus eliminating any intermediate transfers. Finally, if your source documents already exist on the server that is hosting PrizmDoc Server then providing the local path to each file is a good option.

The following sections provide some specific information about each transfer method. Additional information can also be found in the [Viewing Sessions](#) topic.

### HTTP PUT Request

The PUT request is considered the default method of transferring source documents to PrizmDoc Server. It provides the most independent method of transferring documents because the data is sent directly to PrizmDoc Server, largely reducing the chances of a failure as compared to the other methods.

The request has the following format:

#### Example

```
PUT ViewingSession/u{ViewingSessionID}/SourceFile?FileExtension={ext}
```

The document data should be included directly in the request body. This request should be made after the HTTP POST request to create a viewing session and before the HTTP POST to start a viewing session.

### HTTP URL

PrizmDoc Server supports downloading a document from an HTTP or HTTPS URL. The URL must either be publicly accessible, or at least accessible over the internal network to which PrizmDoc Server is connected.

The HTTP PUT request is not required if this transfer method. Instead, some JSON properties in the body of the HTTP POST request to create a new viewing session must be set appropriately. These specific

properties set in this request.

To enable the HTTP URL transfer method, set the **documentSource** JSON property to **http**. This will inform PrizmDoc Server to expect a valid URL to be specified by the **externalId** JSON property. Set the **externalId** JSON property to the HTTP URL of your document. PrizmDoc Server will require a valid extension be provided as well. You can set the extension explicitly with the **documentExtension** JSON property. If **documentExtension** is not set, PrizmDoc Server will attempt to extract an extension from the end of the URL resource name. For example:

## Example

```
POST /ViewingSession
{
  "documentSource": "http",
  "externalId": "http://localhost/osha.pdf",
  "documentExtension": "pdf"
}
```

When PrizmDoc Server receives the POST request to create the viewing session, it will begin downloading the document asynchronously in a background thread while the request returns with a status of 200 OK. Because the document download occurs asynchronously within PrizmDoc Server, it is possible that the retrieval process will fail without any immediate feedback. In this case, the viewing session will be stopped, and any request in the context of the same viewing session will return in error with a status code of 580 and HTTP reason of **Document download failed**.

For additional security, there are user-configurable properties in the central configuration file called **viewing.sessionConstraints.externalId.regex** and **viewing.sessionConstraints.documentExtension.regex** that can be used to restrict values coming in from the **externalId** and **documentExtension** JSON properties, respectively. Any values that do not pass these filters will cause the viewing session to be stopped immediately. See the Central Configuration topic for details about this and other configuration properties.

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

There is a user-configurable timeout value that PrizmDoc Server will use when it sends the GET HTTP request to download the document. This timeout value is specified in `pcc.config` under the property **DocumentAcquisitionTimeout**. The default value is 25 seconds. See the PrizmDoc Server [Configuration](#) topic for details about other user-configurable properties.

For additional security, there is a user-configurable property in `pcc.config` called **ViewingSessionPropertyExternalId** that can be used to restrict values coming in from the **externalId** JSON property. Any values that do not pass this filter will cause the viewing session to be stopped immediately. See the PrizmDoc Server [Configuration](#) topic for details about this and other configuration properties.

## Local File

The third option to transfer documents makes use of the local file system to make your documents available to PrizmDoc Server. This option is enabled by setting the **documentSource** JSON property to **file**. This will inform PrizmDoc Server to expect a valid local file name to be specified by the **externalId**

will require a valid extension be provided as well. You can set the extension explicitly with the **documentExtension** JSON property. If **documentExtension** is not set, PrizmDoc Server will attempt to extract an extension from the end of the filename given.

### Example

```
POST /ViewingSession
{
  "documentSource": "file",
  "externalId": "myDocumentStore\\osha.pdf",
  "documentExtension": "pdf"
}
```

For security purposes, only partial/relative paths and filenames are supported in **externalId**. Rooted paths are not supported. Instead, the root directory where your documents will exist should be specified in `pcc.config` using the **UserDocumentFolder** property. PrizmDoc Server will combine the root directory in this property with the value it receives in **externalId**. If, when combined, a relative path in **externalId** takes the current directory outside of the directory specified in **UserDocumentFolder**, or any other invalid path is used, the transfer will fail.

The following are examples of valid values for **externalId**:

- "sample.doc"
- "docFolder\\sample.doc" (Windows)
- "docFolder/sample.doc" (Linux)

When PrizmDoc Server receives the POST request to create the viewing session, it will copy the document asynchronously in a background thread while the request returns with a status of OK. Because the document is copied asynchronously within PrizmDoc Server, it is possible that the retrieval process will fail without any immediate feedback. In this case, the viewing session will be stopped, and any request in the context of the same viewing session will return in error with a status code of 580 and HTTP reason of **Document download failed**.

For additional security, there are user-configurable properties in the central configuration file called **viewing.sessionConstraints.externalId.regex** and **viewing.sessionConstraints.documentExtension.regex** that can be used to restrict values coming in from the **externalId** and **documentExtension** JSON properties, respectively. Any values that do not pass these filters will cause the viewing session to be stopped immediately. See the Central Configuration topic for details about this and other configuration properties.

 The following configuration properties have been deprecated and will be removed in a future release. Alter these properties only if not using the central configuration file.

There is a user-configurable timeout value that PrizmDoc Server will use when it copies the document. This timeout value is specified in `pcc.config` under the property **DocumentAcquisitionTimeout**. The default value is 25 seconds. See the PrizmDoc Server [Configuration](#) topic for details about other user-configurable properties.

For additional security, there is a user-configurable property in `pcc.config` called **ViewingSessionPropertyExternalId** that can be used to restrict values coming in from the **externalId** JSON property. Any values that do not pass this filter will cause the viewing session to be stopped

properties.

## Use a Viewing Session

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 Application Services](#).

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

```
POST http://localhost:18681/PCCIS/V1/ViewingSession
Content-Type: application/json
{
  "render": {
    "html5": {
      "alwaysUseRaster": false
    }
  }
}
```

#### Example Response

```
200 OK
Content-Type: application/json
{
  "viewingSessionId": "-
gchUEYvBE50CgcWJajgoXcW7QD0I8zNDFlexD9hzbXkmrYlw8DrxJ-
KiHAf2oTAL_HiHK1MsstBlNgZFCrcJQ"
}
```

### Step 2: (Optional) Upload the Source Document

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

```
PUT http://localhost:18681/PCCIS/V1/ViewingSession/u-
gchUEYvBE50CgcWJajgoXcW7QD0I8zNDFlexD9hzbXkmrYlw8DrxJ-
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

```
POST http://localhost:18681/PCCIS/V1/ViewingSession/u-
gchUEYvBE50CgcWJajgoXcW7QD0I8zNDFlexD9hzbXkmrYlw8DrxJ-
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

```
GET http://localhost:18681/PCCIS/V1/Page/q/0?DocumentID=u-
gchUEYvBE50CgcWJajgoXcW7QD0I8zNDFlexD9hzbXkmrYlw8DrxJ-
KiHAf2oTAL_HiHK1MsstBlNgZFCrcJQ&ContentType=png
```

### Example Response

```
200 OK

Content-Type: image/png

{image data}
```

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

```
POST http://localhost:18681/PCCIS/V1/ViewingSession/u-
gchUEYvBE50CgcWJajgoXcW7QD0I8zNDFlexD9hzbXkmrYlw8DrxJ-
KiHAf2oTAL_HiHK1MsstBlNgzFCrcJQ/Notification/SessionStopped
```

## Example Response

```
200 OK
```

## Watermark Content in a Viewing Session

The PrizmDoc Back-end RESTful 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.

 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.

 Watermarks are currently fully supported only when viewing SVG content. When viewing watermarked raster content, multi-line text, text decoration, and complex Unicode fonts such as those used for Hebrew and Arabic are not supported. Watermarks using these options will not appear when viewing is performed in the Internet Explorer 8 browser, which does not support SVG.

## 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.

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 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 API does not define a limit for the watermark image size or resolution.

### Source Locations

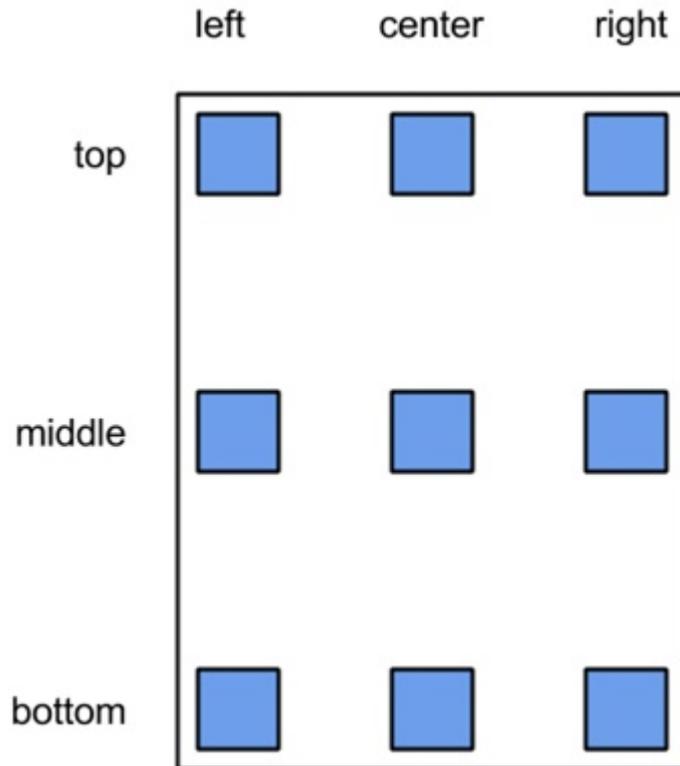
The source of image watermarks can be either a URL or work file ID.

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 Back-end RESTful Services are running.

See the [Work Files](#) topic for more information about creating work files and getting their IDs.

## Aligning Watermarks

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": {
```

```
    },  
    "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

**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 \_\_\_\_\_

**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
```

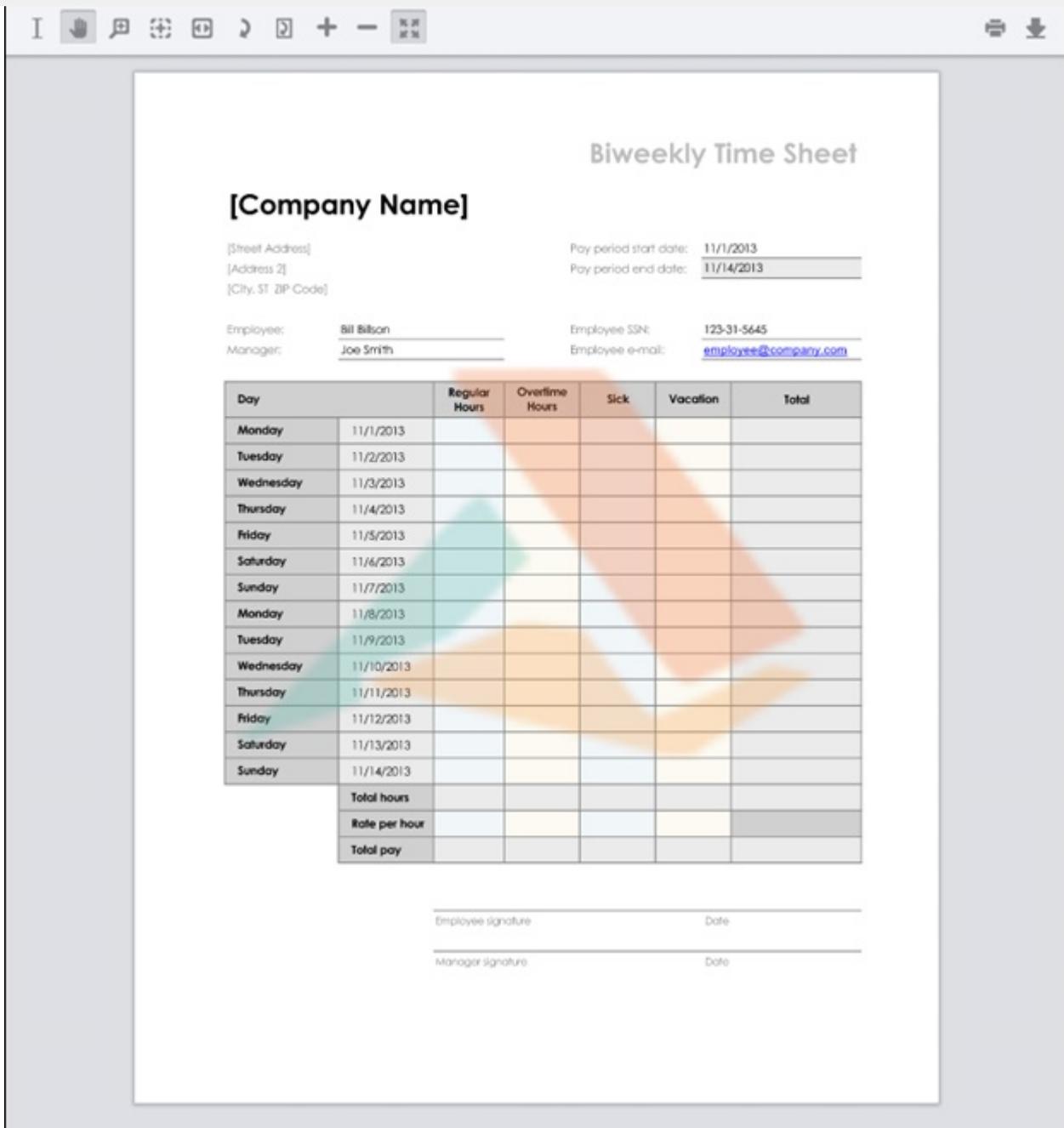
```
    "externalId": "my-application-name",  
    "externalId": "my-unique-document-name.docx",  
    "render": {  
      "html5": {  
        "alwaysUseRaster": false  
      }  
    },  
    "watermarks": [  
      {  
        "type": "diagonalText",  
        "slope" : "up",  
        "opacity": 0.25,  
        "text": "Accusoft\nConfidential",  
        "fontSize": "50pt",  
        "color": "#FF0000",  
        "fontWeight": "bold",  
        "fontFamily": "Arial"  
      }  
    ]  
  }  
}
```

Sample Output



```
renderer: {  
  "html5": {  
    "alwaysUseRaster": false  
  }  
},  
"watermarks": [  
  {  
    "type": "image",  
    "opacity": 0.3,  
    "src": "http://localhost/watermark_images/logo.png",  
    "autoSize": "fit"  
  }  
]  
}
```

Sample Output



## 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",
```

```
    "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					
<b>Total hours</b>						
<b>Rate per hour</b>						
<b>Total pay</b>						

Employee signature \_\_\_\_\_ Date \_\_\_\_\_  
Manager signature \_\_\_\_\_ Date \_\_\_\_\_

**jdoe**  
**11/13/2014 2:24 PM**  
**NOT FOR DISTRIBUTION**

## 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

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.

### Overview

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

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 "parselcons" 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

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

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

Similarly 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

The E-Signature viewers (Template Designer and E-Signer) can be configured in one of three ways as described below.

You can configure:

- The Viewer parameters - [Working with E-Signing > Developer Reference - E-Signing > External: jQuery.fn](#)
- The Viewer control parameters - [Working with the Viewer > Developer Reference - Viewer > ViewerControl > ViewerControlOptions](#)

 If you specify the `documentId` viewer control parameter, it is still necessary to specify the `templateDocumentId` viewer parameter.

### Option 1 - Configure the options that will be set when the Viewer is built

You can edit the `sample-config.js` module file and build the Viewer. This file is located in the `modules/common` folder of the Template Designer sample and E-Signer sample, which are installed when you install PrizmDoc.

### Option 2 - Configure the options without having to build the Viewer

Using PrizmDoc v10.3 or later, in JavaScript you can 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\.net\cs\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 close.

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

Using PrizmDoc v11.1 or later you can disable default embedding of the Viewer and instead use your own code to embed the Viewer into a web page. In order to do this you will need to perform the following steps:

1. Change the **id property** on the div reserved for embedding to something other than the default "pcc-viewer"; doing so will disable auto-embedding.
2. Create a **separate js 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 the **js file** you created in Step #2 above 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>
  <!-- this file will contain code for custom embedding -->
  <script src="viewer-assets/js/embed.js"></script>
  ...
</head>
<body>
  <div id="pcc-viewer-custom"></div>
</body>
```

## Build the E-Signature Viewers

of these viewers you will need to build them. To build either of these viewers, follow the steps below.

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\PCCIS\net\cs\template-designer-sample
```



By default, the C# template designer sample is installed to  
C:\Prizm\Samples\PCCIS\net\cs\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:
  - To create a single developer build, use the following command:
    - **gulp build**
  - 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**
  - 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**
  - 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

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:

```
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
    });
  }
});
```

## End User Guides

The PrizmDoc Viewer is a web application, running in your web browser, that allows you to view and annotate most documents, including Microsoft Office and Adobe PDF, without requiring these applications to be installed on your device.

PrizmDoc is designed to run on every type of computing device, from desktops and laptops to tablets/iPads and even mobile phones - anything with a modern web browser should work. The design of the user interface will adapt to the different sized screens and input methods (mouse, touch) across each of these devices, making it easy to work where and how you wish.

The following sections contain detailed information about the various Viewer features available in PrizmDoc and how to use them to increase your productivity:

- **The Viewer** - The PrizmDoc Viewer is a web application, running in your web browser, that allows you to view and annotate most documents.
- **E-Signature** - The E-Signature module opens up a set of PrizmDoc workflows based on forms stored in file formats supported by PrizmDoc. As with the PrizmDoc Viewer, the E-Signature module works in your website to empower your customers as you see fit.
- **Local File Viewer** - The Local File Viewer is a Windows application that works in conjunction with PrizmDoc to allow you to open files on your local machine in a web browser.

## Viewer Guide

This section contains the following information:

- **Overview**
- **How To**
  - **View Documents**
    - Use Text Selection Options
    - View Embedded Hyperlinks
    - Work with Thumbnails
    - Work with Email Attachments
  - **Search Documents**
    - Search Box and Icons
    - Search Patterns, Filters & Document Review Features
    - Use the Fixed Search Terms Feature
    - Use the Proximity Search Feature
    - Redact Search Results
  - **Annotate Documents**
    - Create Image Stamp Annotations
    - Create a Polyline Annotation
    - Create a Strikethrough Annotation

- Use Annotation Layers
  - Work with Annotation Layers
  - Load Annotations
  - Save Annotations
- Redact Documents
  - Create a Full Page Redaction
  - Use Redaction Reasons
- Use the Comment Feature
  - Use Annotation Comments
  - Use E-Signature Comments
  - Use Redaction Comments
  - Use Skinny Comments
- Print Documents
  - Print Non-standard Size Documents
- Download Documents
  - Downloading the Original Document
  - Burning Annotated Content in the Viewer
  - Burning Redacted Content in the Viewer
  - Burning Signed Content in the Viewer
- Reference
  - Tools
    - Viewer Main Screen Options
    - View Tab
    - Search Tab
    - Annotate Tab
    - Redact Tab
    - E-Sign Tab
    - Keyboard Shortcuts
    - Work with Sticky Mouse Tools

## Overview

### Overview

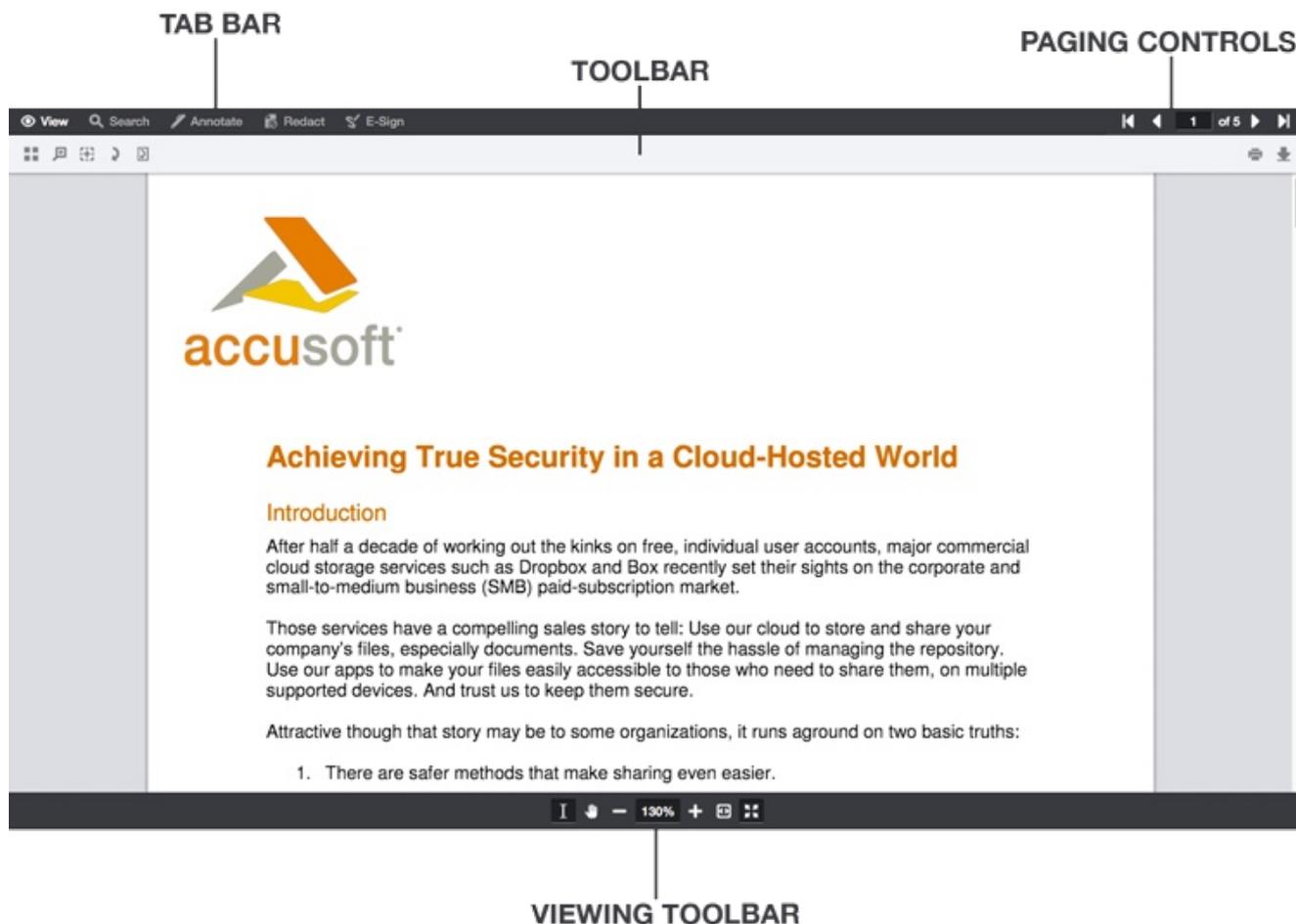
This section provides an overview of the Viewer so you can learn about using the functionality quickly and easily.

- [What is the PrizmDoc Viewer?](#)
- [Understanding the User Interface](#)
- [Viewing Documents](#)
- [Interacting with Documents](#)

The PrizmDoc Viewer is a web application that lets you view, search, and annotate a large number of document types, including Microsoft Office, Adobe PDF, and more. The Viewer is designed to work on any device with a web browser, including desktop and laptop PCs, tablets and smartphones, and will work using a keyboard and mouse or touch input, all without installing a dedicated app.

## Understanding the User Interface

Here is an example of how the Viewer looks:



## Viewing Documents

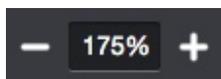
Once a document is loaded in the Viewer, you have the standard features for reading a document, as well as some more advanced features. You can scroll freely through the document, navigate between pages using paging controls, select text, and zoom in and out using a variety of tools, so that you can examine the document in greater detail.

## Moving Between Pages



The Viewer's paging controls let you move between pages. Clicking the buttons on the left and right will take you to the first and last page, respectively. The other two buttons will take you back and forth one page at a time. If

### Zooming & Panning



To zoom in or out, press the plus and minus buttons at the bottom of the Viewer. If you need to zoom at a larger increment, press the zoom percentage to see a list of zoom levels:



Choose from the list of percentages, or choose Full Width (zoom to the page width), Full Height (zoom to the full page height), and Full Page (make the entire page fit in the Viewer).

For very precise zooming, the Viewer has Magnifier and Rectangle Zoom tools:



The Magnifier tool acts like a magnifying glass on a paper document. Selecting this tool, and pressing within the document will magnify that portion of it to see more detail. The Rectangle Zoom tool enables you to press and drag a rectangle around a portion of the document you want to zoom in on. This lets you choose the exact portion of the document you want to focus on.

The Pan tool lets you move the view in a different direction, which is very helpful when you're zoomed in and can't see your whole document. The Pan tool is located at the bottom of the viewer next to the zoom controls:



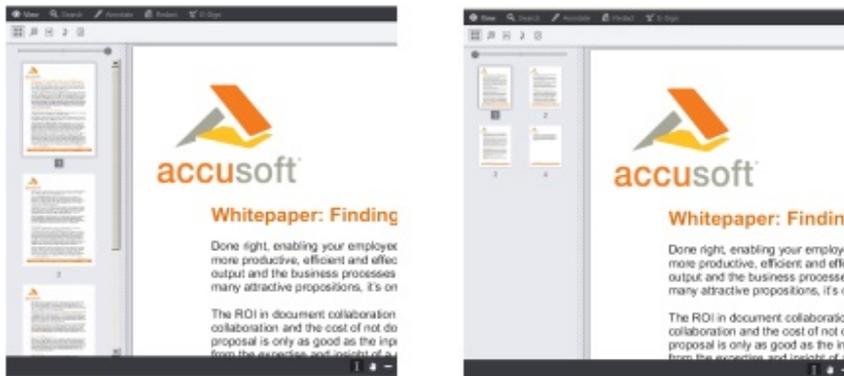
With the Pan tool active, press and drag the cursor in any direction to pan around the document area.

### Page Thumbnails

To see an overview of your document, you can open the Thumbnails panel in the View tab to see small versions of



Once this panel is open, you can scroll through all of the pages, and press a thumbnail to immediately jump to that page. There's a slider at the top of the panel that will increase or decrease the size of the thumbnails if you would like to see more or less detail in the thumbnails:



## Interacting with Documents

In addition to being able to view documents, you are also able to draw annotations, redact content, add e-signatures, download, and print documents all without altering the original.

### Selecting Text

The Text Selection tool allows you to select text and perform common operations on your selection via the Immediate Action Menu (IAM).



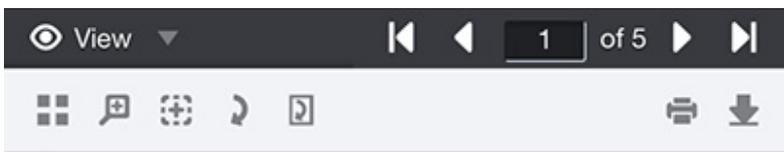
Once selected, press and drag the cursor to select text on a page. When you're finished, release the cursor. If you're using a mouse, a small rectangle with a menu icon appears:

Done right, enabling your employees to collaborate on documents cuts costs by making them more productive, efficient and effective collectively, in addition to improving the quality of their output and the business processes that output supports. It's an attractive proposition, and like many attractive propositions, it's on the other side of a minefield.

Hover over this small rectangle, and you will see a list of common actions to perform with this text like Copy, Highlight, Redact, and more:



On touch devices, when you release the cursor the IAM will appear at the bottom automatically:



oft

## aving True Security in a Cloud-Hosted World

### ction

f a decade of working out the kinks on free, individual user accounts, major commercial rge services such as Dropbox and Box recently set their sights on the corporate and medium business (SMB) paid-subscription market.

ervice have a compelling sales story to tell: Use our cloud to store and share your /s files, especially documents. Save yourself the hassle of managing the repository. apps to make your files easily accessible to those who need to share them, on multiple id devices. And trust us to keep them secure.

though that story may be to some organizations, it runs aground on two basic truths:

here are safe

to matter how o escaping th ocument files

tepaper exam and sharing, a cal storage a

### ility and Cl

lling out to the ne establishin

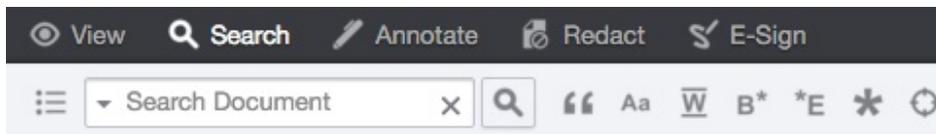
i most commel as the symb failures. The Agency (NSA



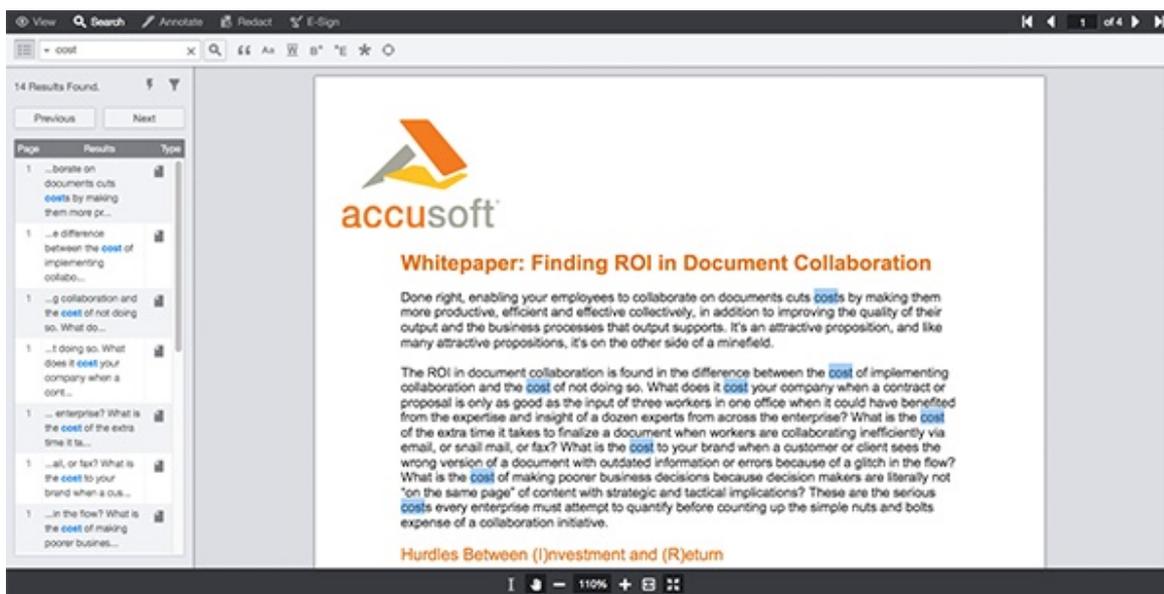
You can choose one of the actions, or press Cancel at the bottom to dismiss the menu.

You will also notice blue text selection handles around the text. If you would like to modify your selection, these handles can be dragged separately to change the selected text.

The Viewer contains a powerful search feature that will let you find words, phrases, and common bits of content like email addresses and postal codes. Opening the Search tab will display a search input box and search operators like match whole word, match case, and proximity search:



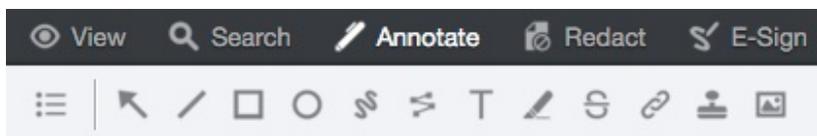
Enter a search term into the input box, and press the Search button or press Enter. The search results panel will open, displaying the search results. The search term(s) will also be highlighted on the document itself:



For more detailed information on searching, refer to the topic: [Working with PrizmDoc > Viewer End-User Guides > Viewer Guide > Search Tab](#).

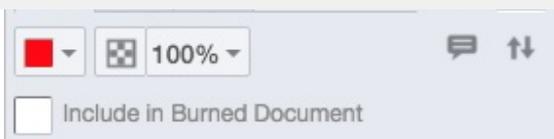
## Annotating Documents

The Viewer allows you to mark up documents using three different categories of annotations: drawing, text, and image. Opening the Annotate tab will show you all various annotation types you can use:



Using these tools, you can add a variety of shapes, highlight text, create links to websites, add images, and more. To add an annotation, press the button for the mark you would like to add. Depending on which tool you've chosen, you might press and drag to add a shape anywhere on the document and of the size you need, or you might select a block of text to add a strikethrough.

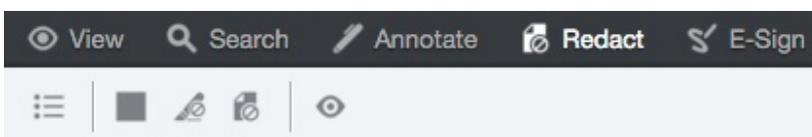
Once you have added an annotation, a context menu will appear letting you change different settings for that mark:



Depending on the mark type, you can change things like color, border thickness, opacity, and layer order, or remove it from the document.

## Redacting Content

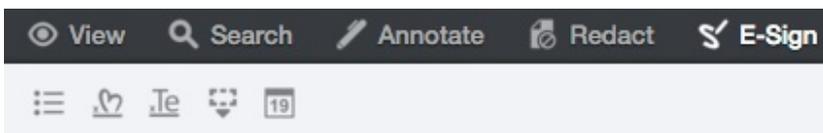
For those working with sensitive content, the Viewer's ability to redact content is a powerful feature. More than simply hiding content visually, once a document with redactions is exported or "burned," the redacted content is securely removed from the resulting document. Opening the Redact tab gives you access to three different ways to create redactions:



You may draw redactions of any size you need, create redactions from selected text, or add redactions to an entire page or range of pages, depending on your needs. Like annotations, redactions have a context menu that gives you access to settings like fill color, border color, and the chance to add text giving the reason why content has been redacted.

## E-Signing

The Viewer's E-Sign tab allows you to sign, initial, and date documents electronically for convenient signing of important documents, removing the need to print, sign, scan, or fax paper copies:



You can draw your signature or initials for use in a document, or type it and choose a stylized font for your signature and initials. All signatures can be saved for use later, and reused among many documents, allowing for fast and easy e-signing. Once created, you can add a signature to the document, and size it and place it anywhere a signature is required. You can also quickly add today's date with the Date tool:

Applicant's Signature: James A. Smith Date: 03/15/2016

## Commenting

After annotations, redactions or e-signatures (collectively known as "marks") have been added to a document in the Viewer, it's possible to add, reply, and view comments associated with them.

### Adding Comments

To add comments to marks, select the object (whether it be an annotation, redaction, or e-signature) to show the context menu. Pressing the comment button on the context menu will open the add comment dialog. Type into the input field, and press Done to add your comment (or cancel if you no longer want to add a comment):

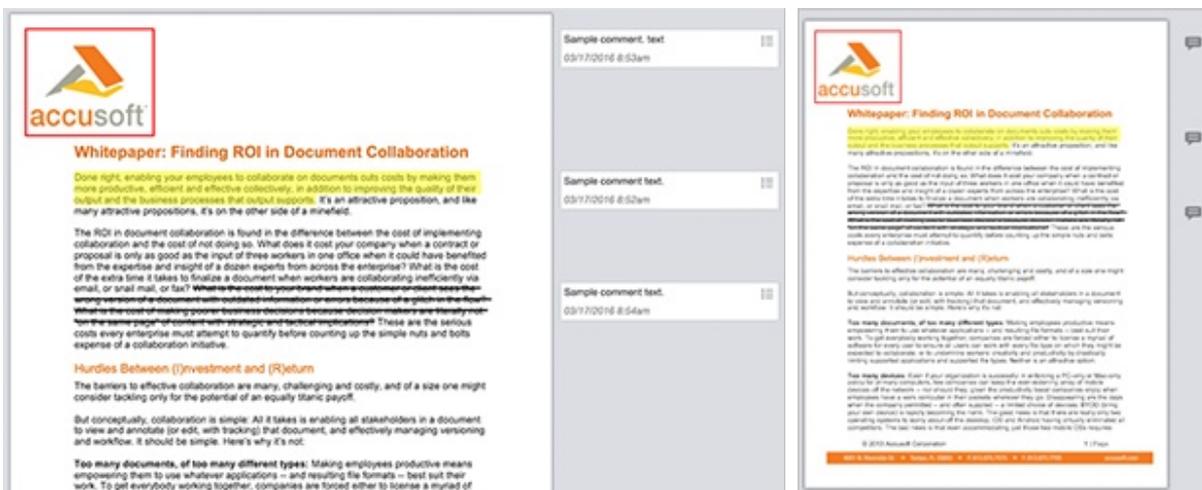


## Viewing Comments

To see the comments you or others have added to a document, press the comments button in the toolbar:



Depending on your screen size or your application's configuration, you will see all comments displayed in full, or in "skinny" mode, where comments are indicated by an icon and are shown only when the indicator is pressed:



To hide the comments, press the comments button in the toolbar again.

## Printing

Documents can be printed from the Viewer with standard printing options like page size and orientation, as well as options for including annotations, redactions and comments. To start the printing process, press the Print button in the toolbar:



Standard printing options like page size and which pages to print are shown first. If you would like to see additional options, press the More options button:

Pages

Print All

Print Current

1-5, 8, 10-15

Show...

Annotations

Redactions

Review Redacted Text

Less options ▾

Orientation

Portrait

Paper Size

Letter

Auto Fit Margins

Comments

Do Not Print

Redaction Reasons

Do Not Print

Print

Select any of the options you need, and press the Print button. The Viewer will prepare the document for printing and launch your web browser's print window to finish the process.

## Downloading

To download documents, you can press the Download button in the toolbar to open the download panel:



In the download panel, you can choose which document format you want to download, which marks you would like to include (if any), and the chance to preview what the downloaded document will look like:

Download as...

PDF

Include...

Annotations All

Redactions

E-Signatures

Preview Download

Once you have chosen the download options that meet your needs, press the Download button to start the download process. When the new file is ready to download, you will see a Save button. Pressing this button will let you save the new file to your device.

For more details on how to use the Viewer, refer to the section: [Viewer Guide](#).

This section contains detailed information on how to use the Viewer features:

- [View Documents](#)
  - [Use Text Selection Options](#)
  - [View Embedded Hyperlinks](#)
  - [Work with Thumbnails](#)
  - [Work with Email Attachments](#)
- [Search Documents](#)
  - [Search Box and Icons](#)
  - [Search Patterns, Filters & Document Review Features](#)
  - [Use the Fixed Search Terms Feature](#)
  - [Use the Proximity Search Feature](#)
  - [Redact Search Results](#)
- [Annotate Documents](#)
  - [Create Image Stamp Annotations](#)
  - [Create a Polyline Annotation](#)
  - [Create a Strikethrough Annotation](#)
  - [Create a Text Hyperlink Annotation](#)
  - [Use Annotation Layers](#)
    - [Work with Annotation Layers](#)
    - [Load Annotations](#)
    - [Save Annotations](#)
- [Redact Documents](#)
  - [Create a Full Page Redaction](#)
  - [Use Redaction Reasons](#)
- [Use the Comment Feature](#)
  - [Use Annotation Comments](#)
  - [Use E-Signature Comments](#)
  - [Use Redaction Comments](#)
  - [Use Skinny Comments](#)
- [Print Documents](#)
  - [Print Non-standard Size Documents](#)
- [Download Documents](#)
  - [Downloading the Original Document](#)
  - [Burning Annotated Content in the Viewer](#)
  - [Burning Redacted Content in the Viewer](#)
  - [Burning Signed Content in the Viewer](#)

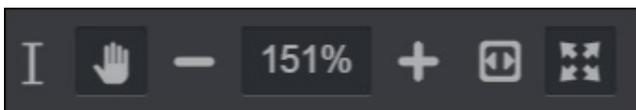
## VIEW DOCUMENTS

This section contains topics that demonstrate the viewing tools including paging, zoom controls, fit options, thumbnails, email attachments, and page rotation:

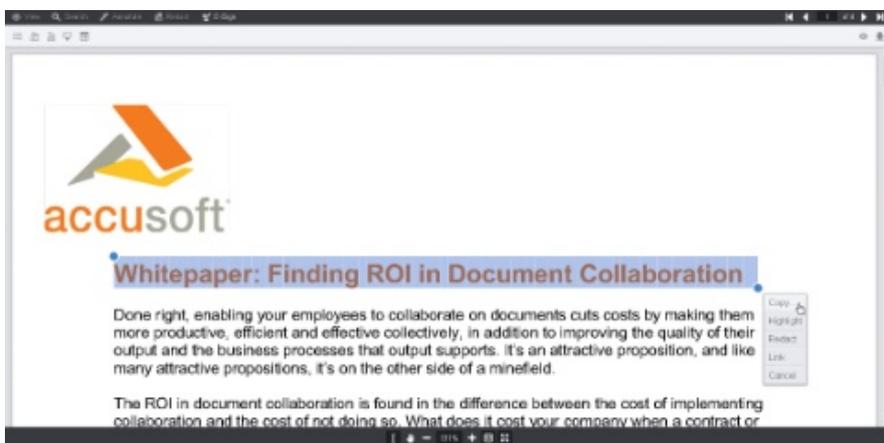
- [Use Text Selection Options](#)
- [View Embedded Hyperlinks](#)
- [Work with Thumbnails](#)
- [Work with Email Attachments](#)

## Use Text Selection Options

The Text Selection Mouse Tool is available at the bottom of the Viewer and allows you to Select Text, Pan, Zoom Out, select viewing options including Percentage, Full Width, Full Height, and Full Page, Zoom In, Fit Content and Full Screen:



Text Selection provides options to select text, Copy, add a Highlight annotation, Redact text, or add a Hyperlink annotation from the Context Menu. You can easily modify your text selection by using either touch or mouse:



Sections in this topic:

- [Adding a Highlight Annotation](#)
- [Redacting Text](#)
- [Adding a Hyperlink Annotation](#)
- [Copying Text \(non-touch device\)](#)
- [Copying Text \(touch device\)](#)
- [Copying Text \(Android device\)](#)

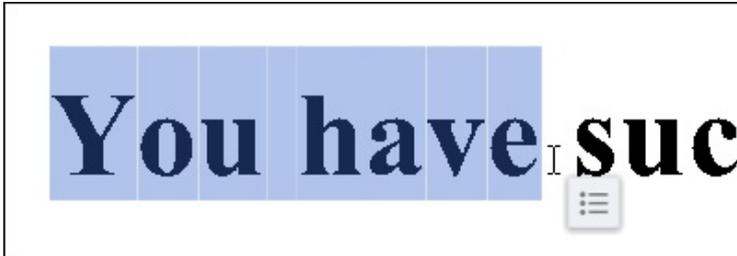
### Adding a Highlight Annotation

To add a Highlight annotation to text in the document:

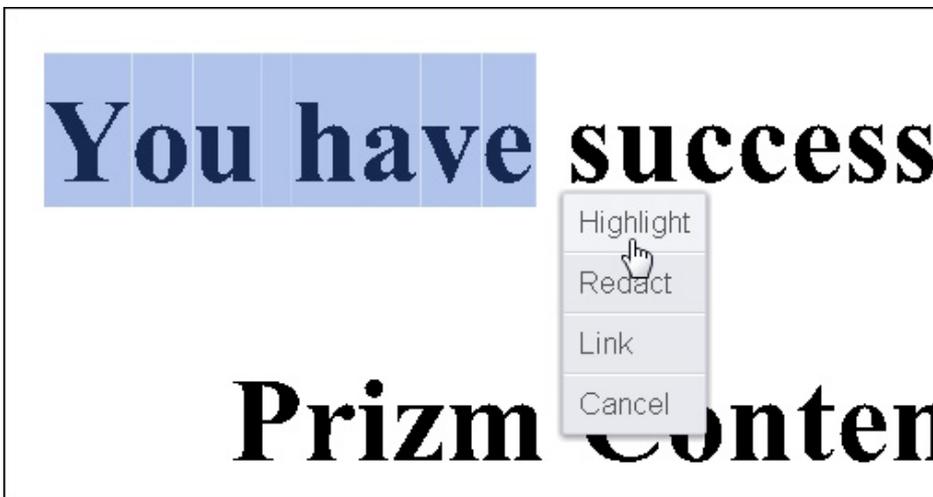
1. Click on the **Text Selection** mouse tool:



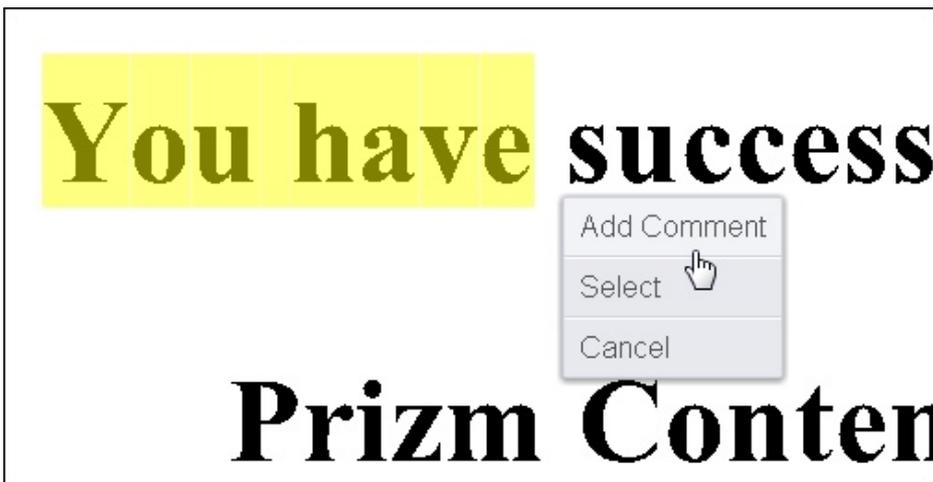
2. Select the **text** you want to highlight and the Context Menu displays:



3. Hover over the **Context Menu** and select **Highlight**:



4. The text in your document is now highlighted. Another Context Menu displays:



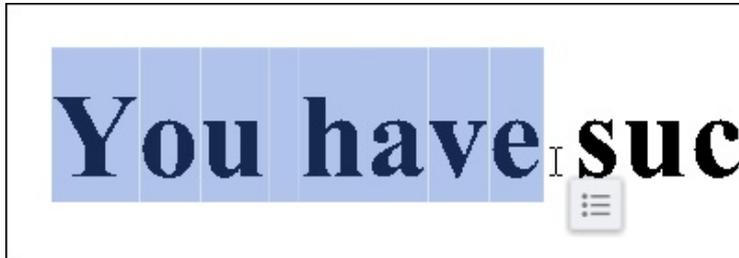
5. You can **Add Comment**, **Select** the annotation or **Cancel**.

## Redacting Text

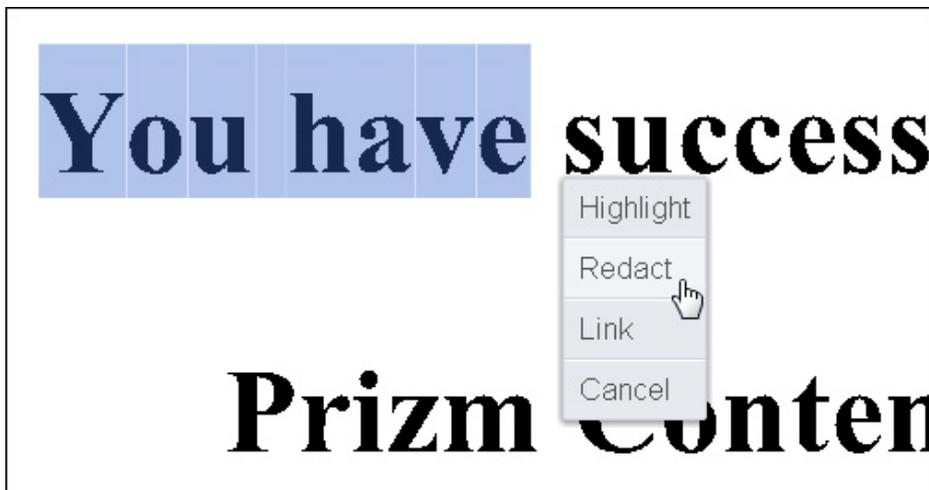
To redact text in the document:

1. Click on the **Text Selection** mouse tool:

2. Select the **text** you want to redact and the Context Menu displays:



3. Hover over the **Context Menu** and select **Redact**:



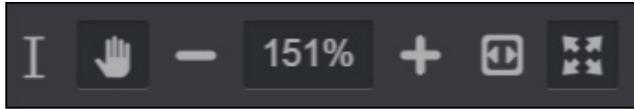
4. The text in your document is now redacted. Another Context Menu displays:



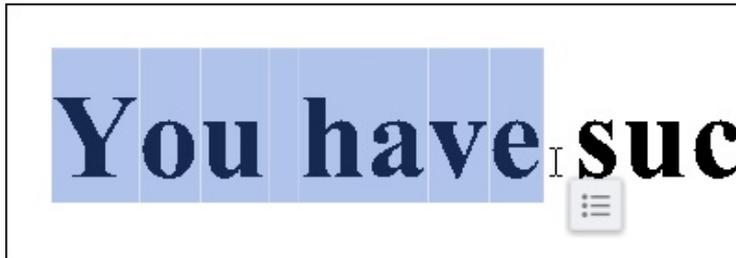
5. You can **Add Comment**, **Select** the redaction, **Cancel** or **Apply a Redaction Reason**.

To add a Hyperlink annotation in the document:

1. Click on the **Text Selection** mouse tool:



2. Select the **text** you want to add a hyperlink to and the Context Menu displays:



3. Hover over the **Context Menu** and select **Link**:



4. The Hyperlink text box displays. Enter the **web address** you want to link to and click the **Checkmark icon**:

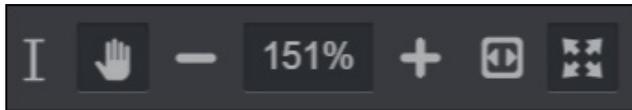


5. Click on the **Hyperlink text annotation** and the Comments Menu displays. You can **Visit the Link**, **Add a Comment**, **Edit the Link**, or **Delete** the annotation:

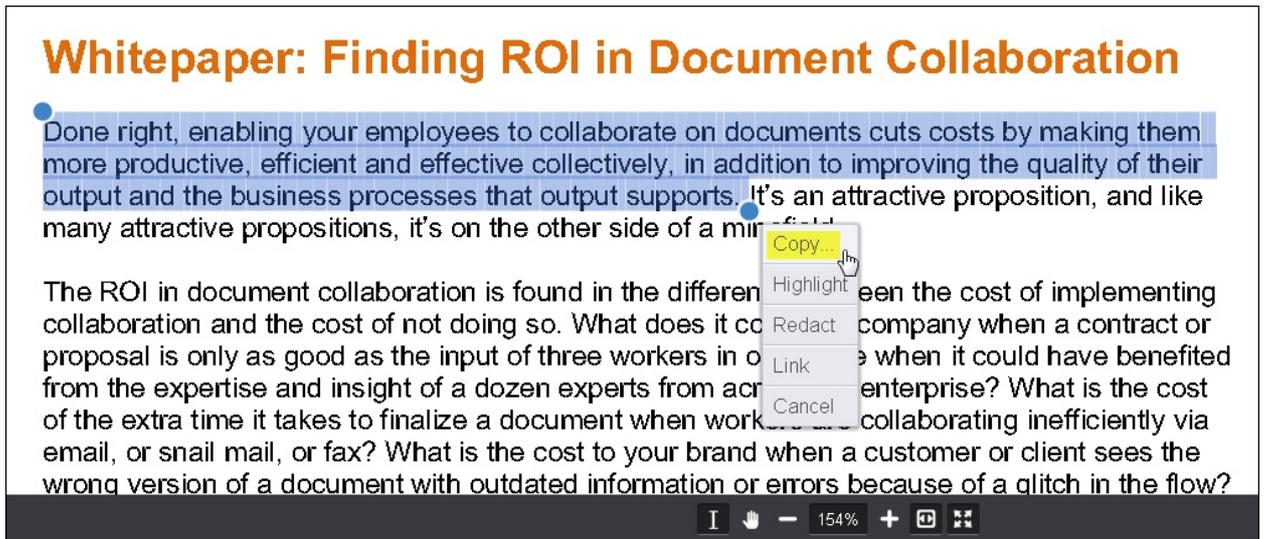


Copying Text (non-touch device)

1. To copy text, use the **Text Selection** tool located under the **View** tab at the bottom of the screen:



2. Select the **text** and the immediate action menu displays. Select **Copy...**



3. Paste the text as desired.

### Copying Text (touch device)

1. Select **text** and the immediate action menu displays:



2. Tap on the "**Copy...**" item.

3. Paste the text as desired.

### Copying Text (Android device)

1. Select **text** and the immediate action menu displays:



2. Tap on the "Copy..." item.
3. Paste the text as desired.

## View Embedded Hyperlinks

Viewing embedded hyperlinks is supported for:

- **External Links** - Office and PDF documents
- **Internal Links** - PDF documents only

- Email addresses
- Intra-document links (to another page within the document)
- External links to websites

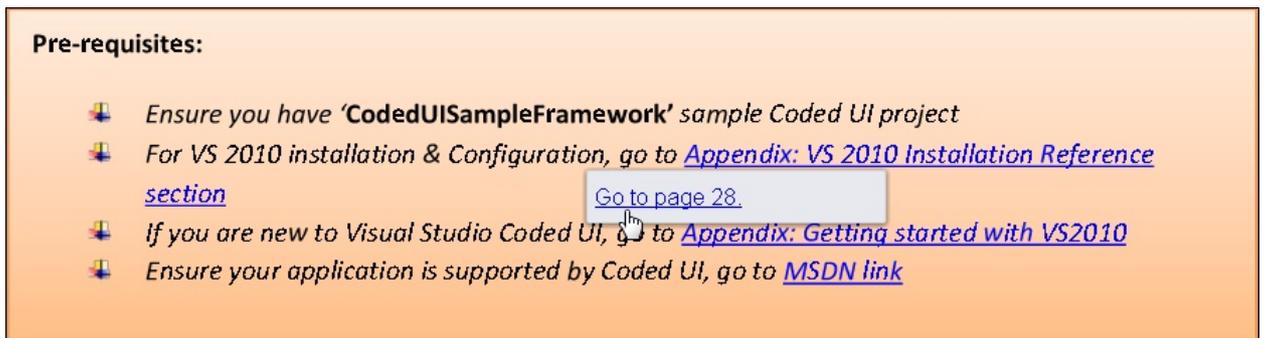
## Email addresses

1. When you click on an **email address**, the 'mail to' box appears and you can click **the link** to launch your email client and send an email:

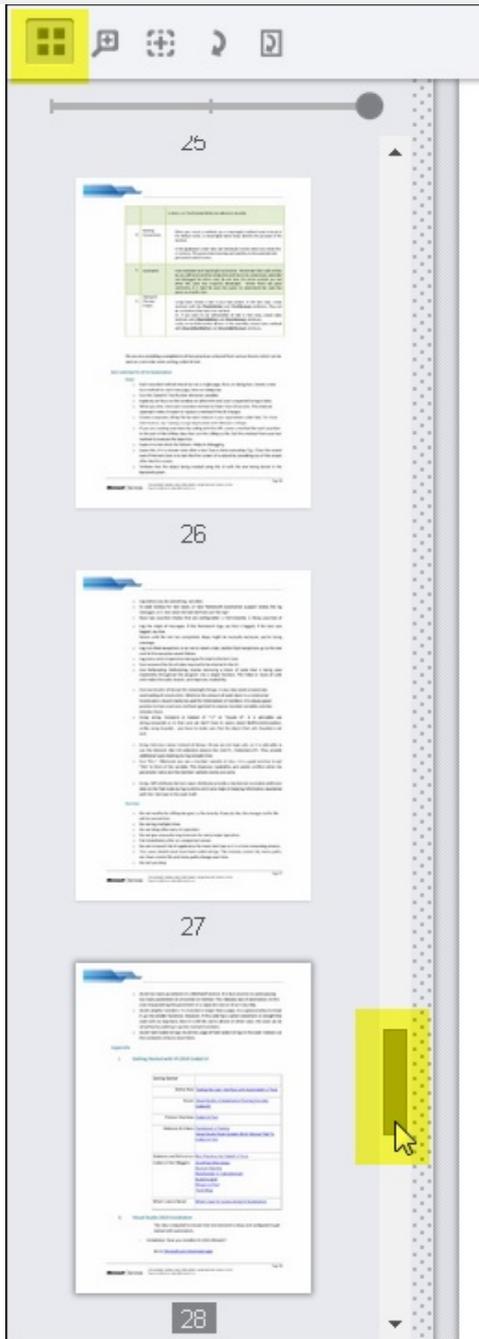


## Intra-document links (to another page within the document)

1. When you click on **the link**, the 'go to page' box appears and you can click on **the link** to go to that page:



2. To return to the original page, click on the **Thumbnail icon** to view the pages within the document and you can easily scroll back to the page you were on:



### External links to websites

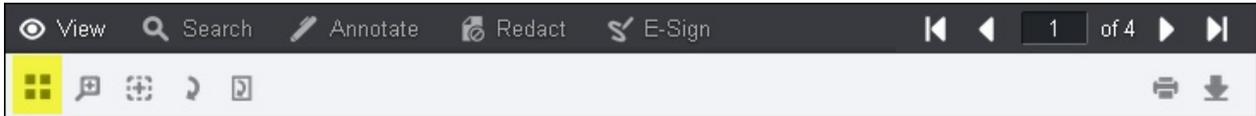
1. When you click on **the link**, the URL shows and you can click on **the link** to go to the website:

Figure 1: The topology required for setting up Build, Deploy and Test environment using various VS components like Build Controller and Agents for Build, Test Controller and Agent for distributing test and collecting results, Virtual environment for Deployment and Execution (Reference taken from [MSDN](http://msdn.microsoft.com/en-us/library/fda2bad5.aspx))

<http://msdn.microsoft.com/en-us/library/fda2bad5.aspx>

## Work with Thumbnails

The Thumbnails icon is located on the View tab and offers several ways to view Thumbnails:



To use the Thumbnails icon for viewing:

1. Click on the **Thumbnails icon**. The Thumbnails pane displays:



2. Click on the **scrollbar** to scroll down and view additional thumbnails:



3. Click on and drag **the slider** to the left to reduce the size of the thumbnails:

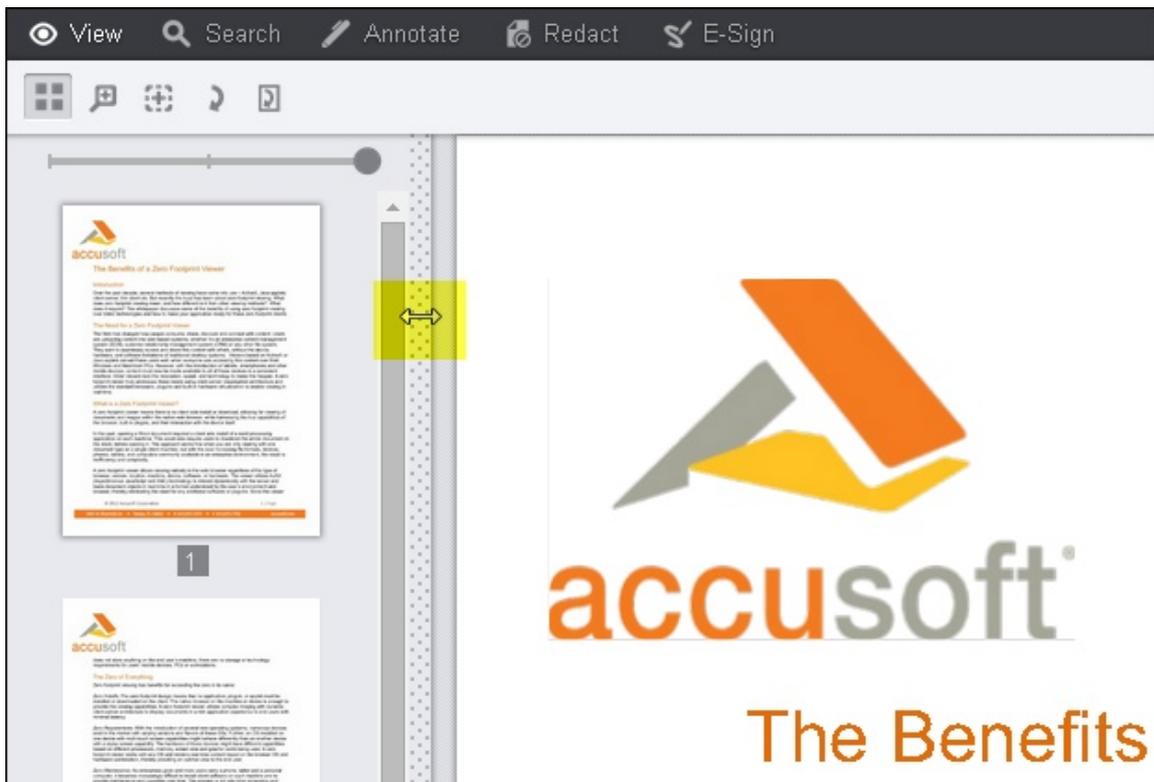


4. To further reduce the size of the thumbnails, click on **the slider** and drag it all the way to the left:

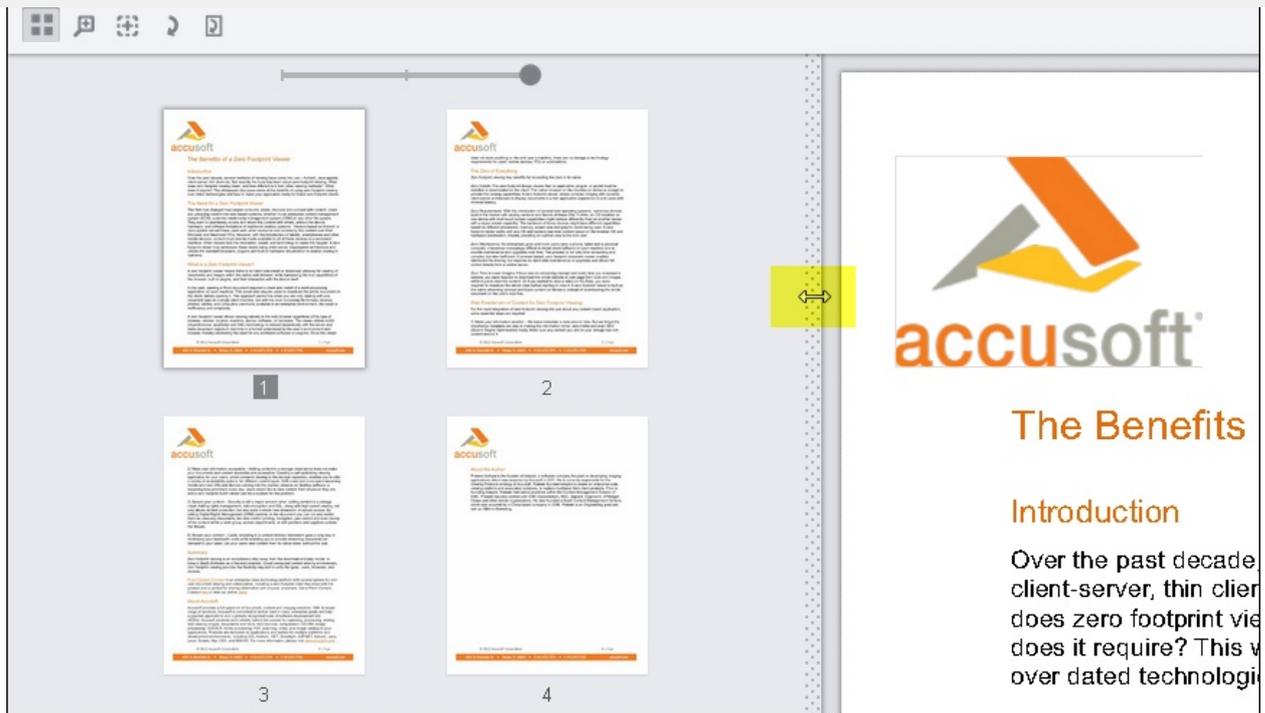


To enlarge the size of the thumbnails, click on **the slider** and drag it all the way to the right.

5. You can also click on the dotted area to resize the Thumbnail pane. Click on the **dotted area** and drag it to the right or left as desired:

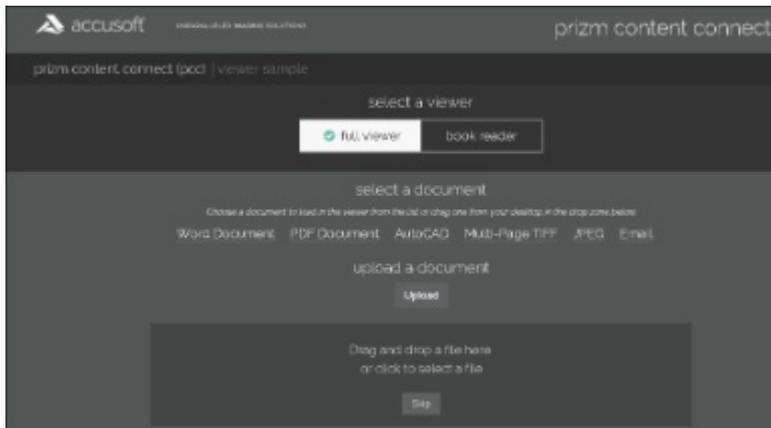


Drag the dotted bar to the right to expand the pane:

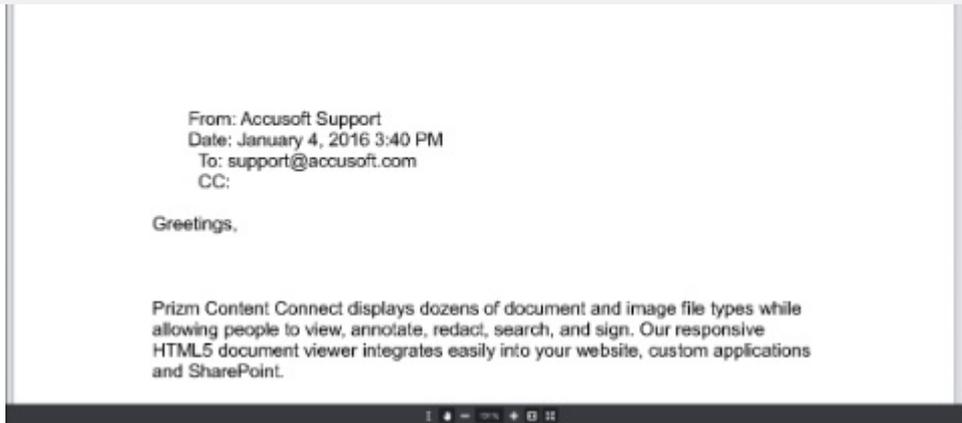


## Work with Email Attachments

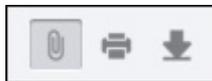
1. To open an email with attachments in the Viewer, select **Email** from the Splash screen:



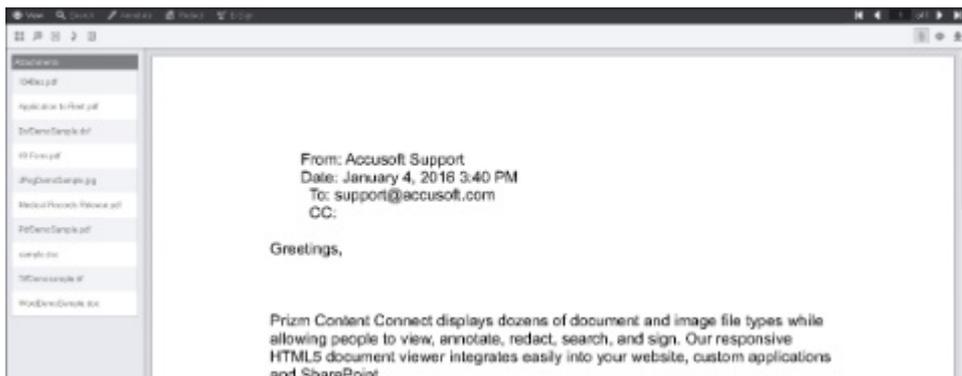
2. the Viewer opens and the email is displayed:



- 3. To see the attachments, click on the **Paper Clip** icon located on the right-hand side of the toolbar:

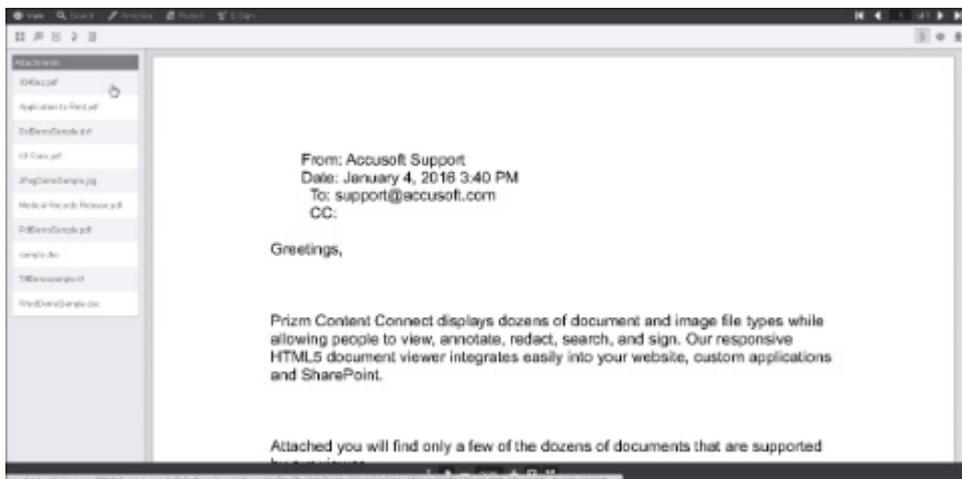


- 4. The attachments are displayed in the panel on the left-side of the Viewer:

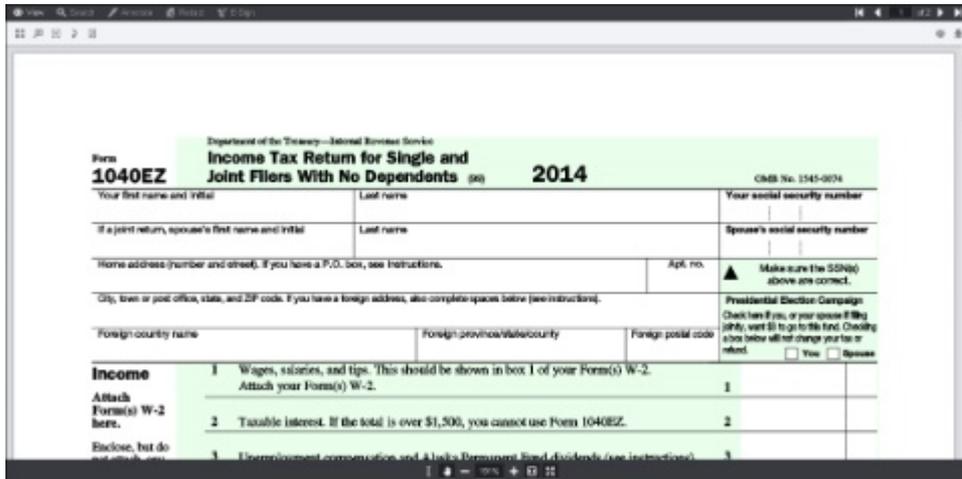


 Note that long file names will wrap inside the panel so you can see the full file name and the document extension.

- 5. To view the attachments, click on the **name of the attachment** you want to view:



browser):



## Search Documents

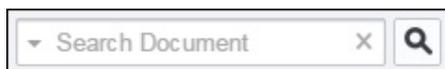
This section explains how to run a simple search on a document and how to access previous and pre-defined searches:

- [Search Box and Icons](#)
- [Search Patterns, Filters & Document Review Features](#)
- [Use the Fixed Search Terms Feature](#)
- [Use the Proximity Search Feature](#)
- [Redact Search Results](#)

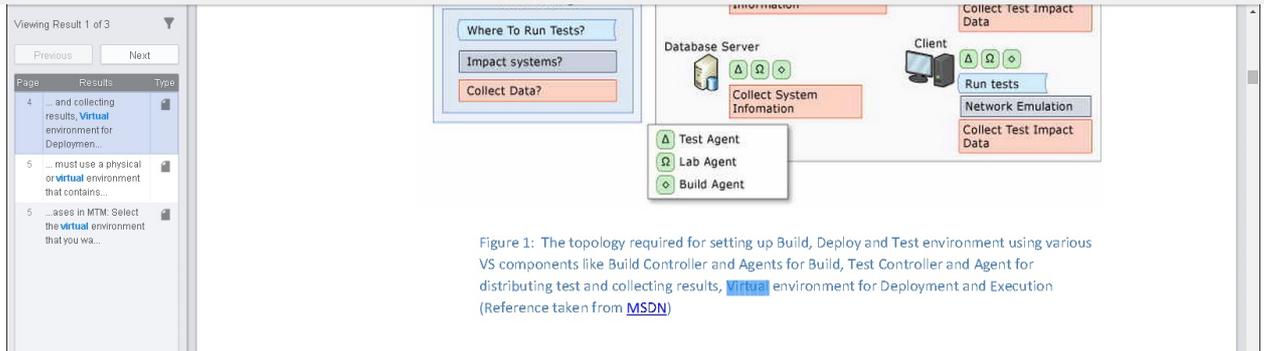
## Search Box and Icons

To use the search document text box:

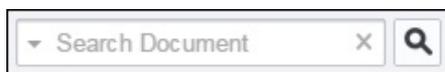
1. Enter the **text** you want to search for in the Search Document text box and click the **Magnifying Glass** (or press **Enter**):



2. The Viewer will return the results listing the page number in the left-hand viewing pane. Click on the **result** you want and the Viewer will display the page in the right-hand pane:

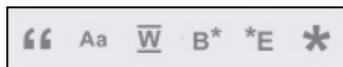


3. To clear the search results, click on the **X** in the Search Document text box:

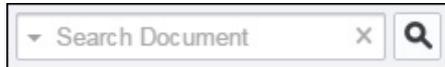


### To use the search icons:

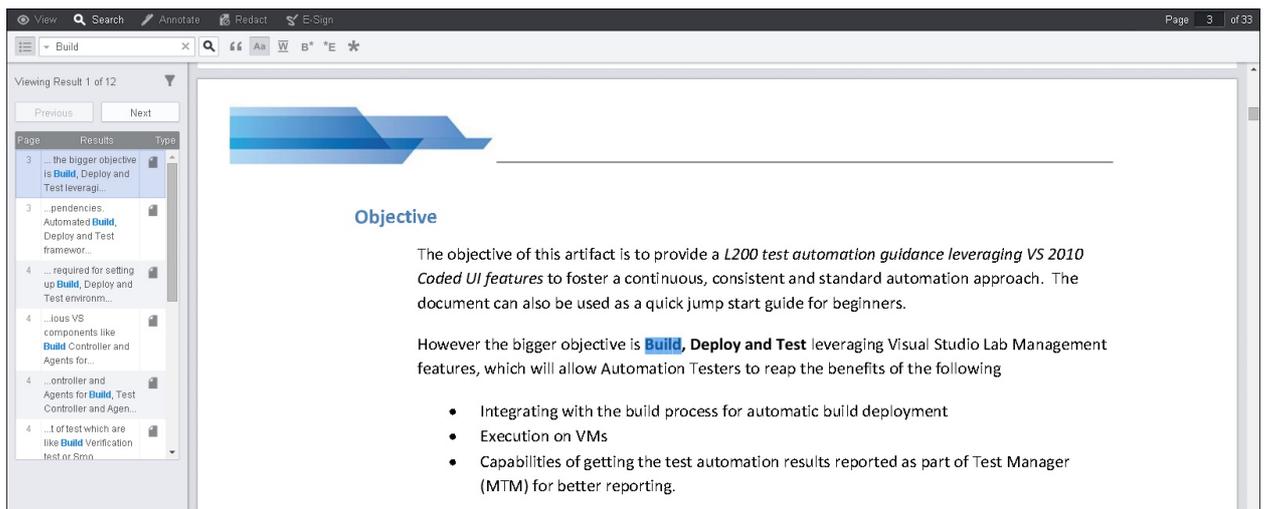
1. Click on the **search icon** you want to use:



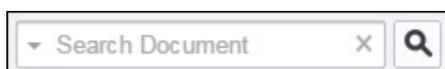
2. Enter the **text** in the Search Document text box and click the **Magnifying Glass** (or press **Enter**):



3. The Viewer returns the results (for example, the **Match Case icon** was selected and the word **Build** was searched for):



4. To clear the search results, click on the **X** in the Search Document text box:



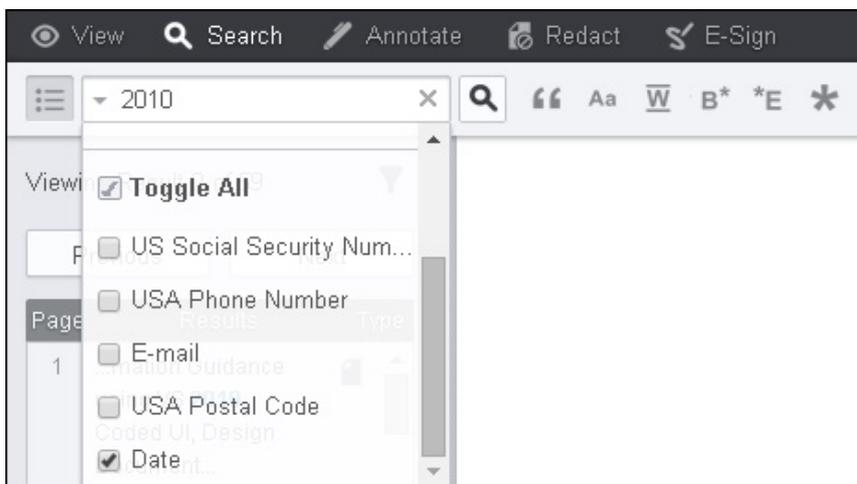
## Features

### Search Patterns

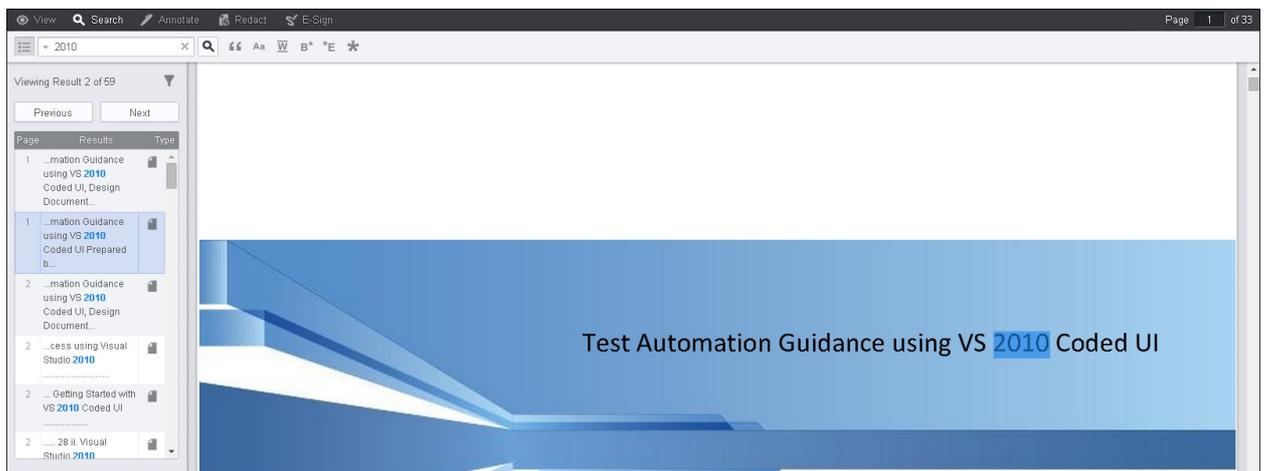
The search patterns can be customized to use a wide variety of regular expressions. See [How to Use Predefined Search](#) for more information.

#### To use the search patterns:

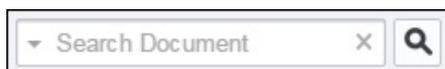
1. Click on the Search Document drop-down arrow and select the **search pattern** you would like to use (for example, click on **Date** and then enter the **date** in the Search Document text box) and click **Search** (or press **Enter**):



2. The Viewer returns the results:



3. To clear the search results, click on the **X** in the Search Document text box:

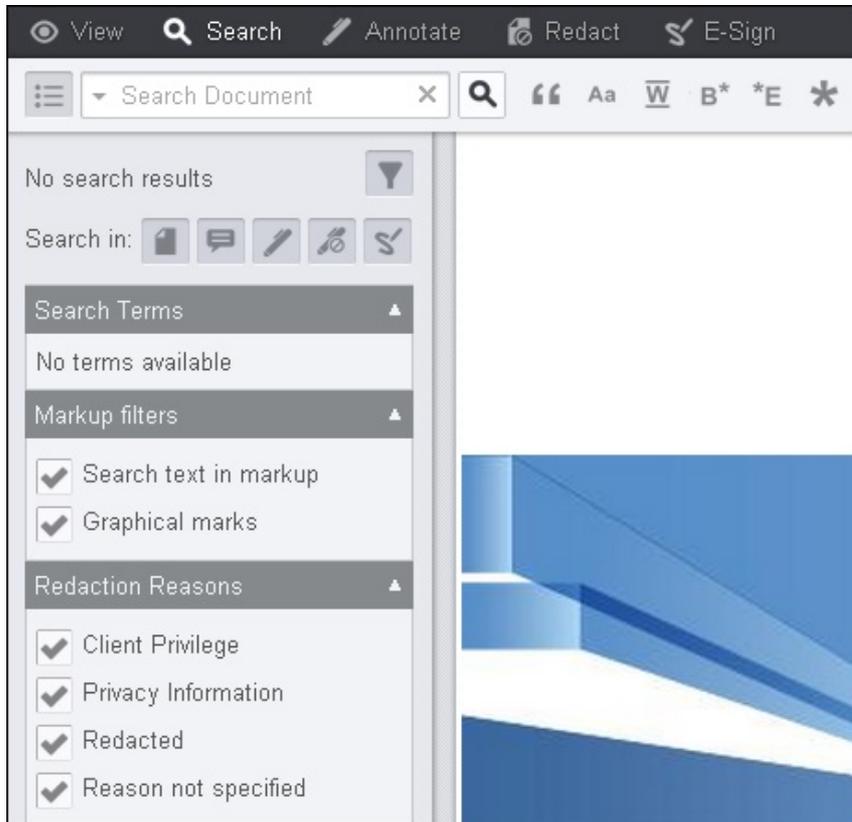


### Search Filters & Document Review Features

#### To use the search filter:

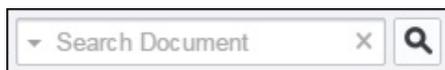
1. Click on the **Search Options** icon:

2. The default **Search Filter** is displayed:

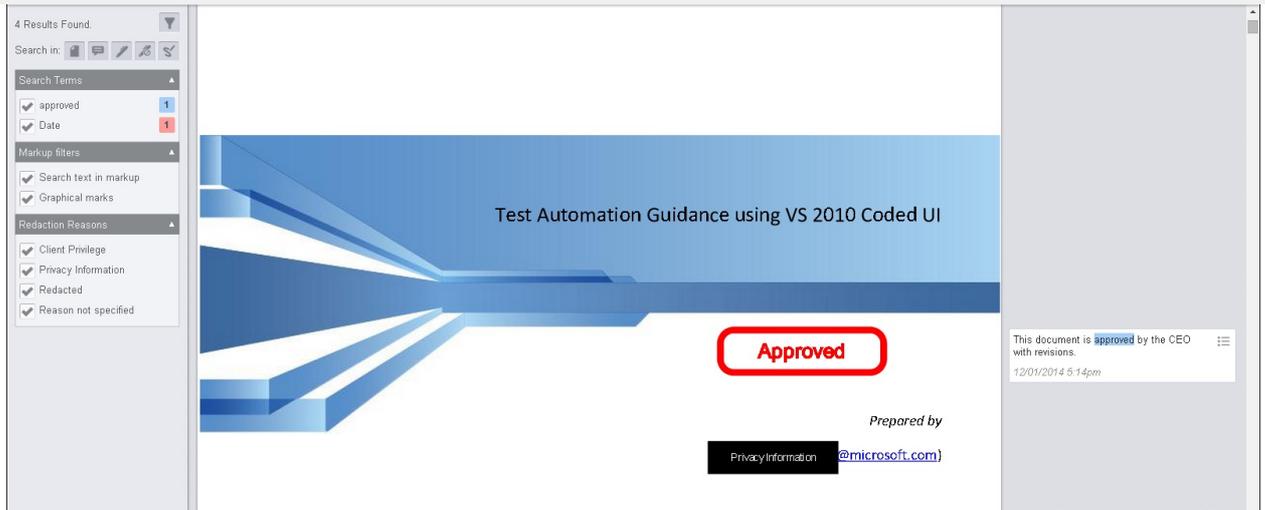


All of the options are selected by default. If you don't want to include one or more of the options in your search, you can deselect them.

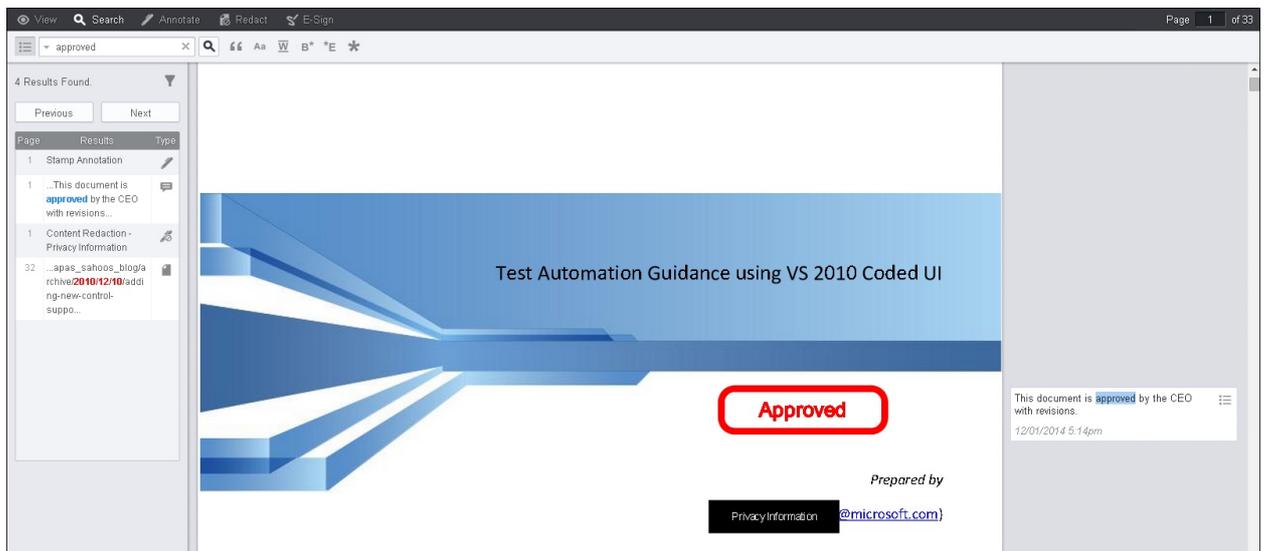
3. Enter the **text** you want to search for in the Search Document text box and click the **Magnifying Glass** (or press **Enter**):



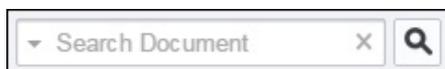
4. In this example, the term "approved" is entered and the Viewer returns two search terms and four results:



5. Click on the filter icon to view the **Page**, the **Results** and the **Type** found:



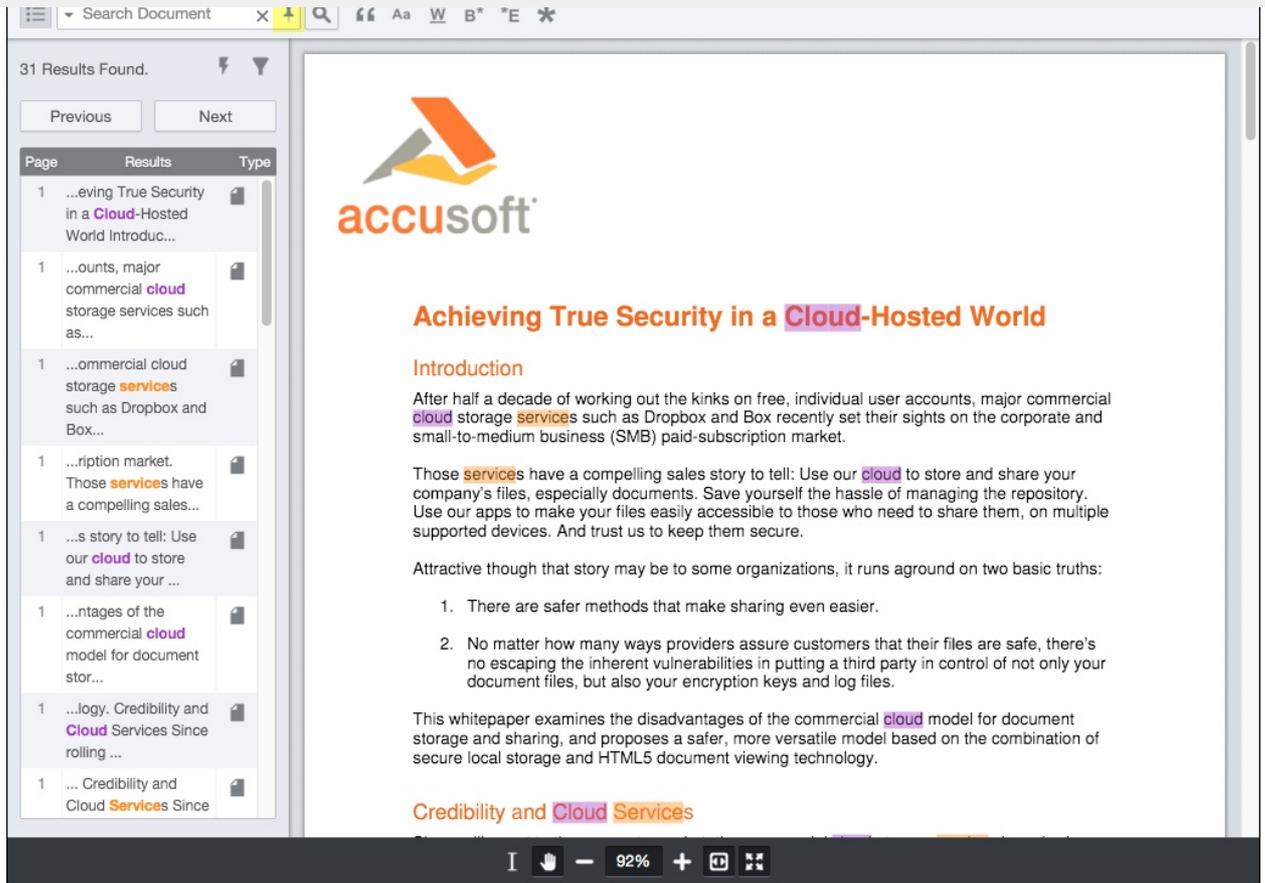
6. To clear the search results, click on the **X** in the Search Document text box:



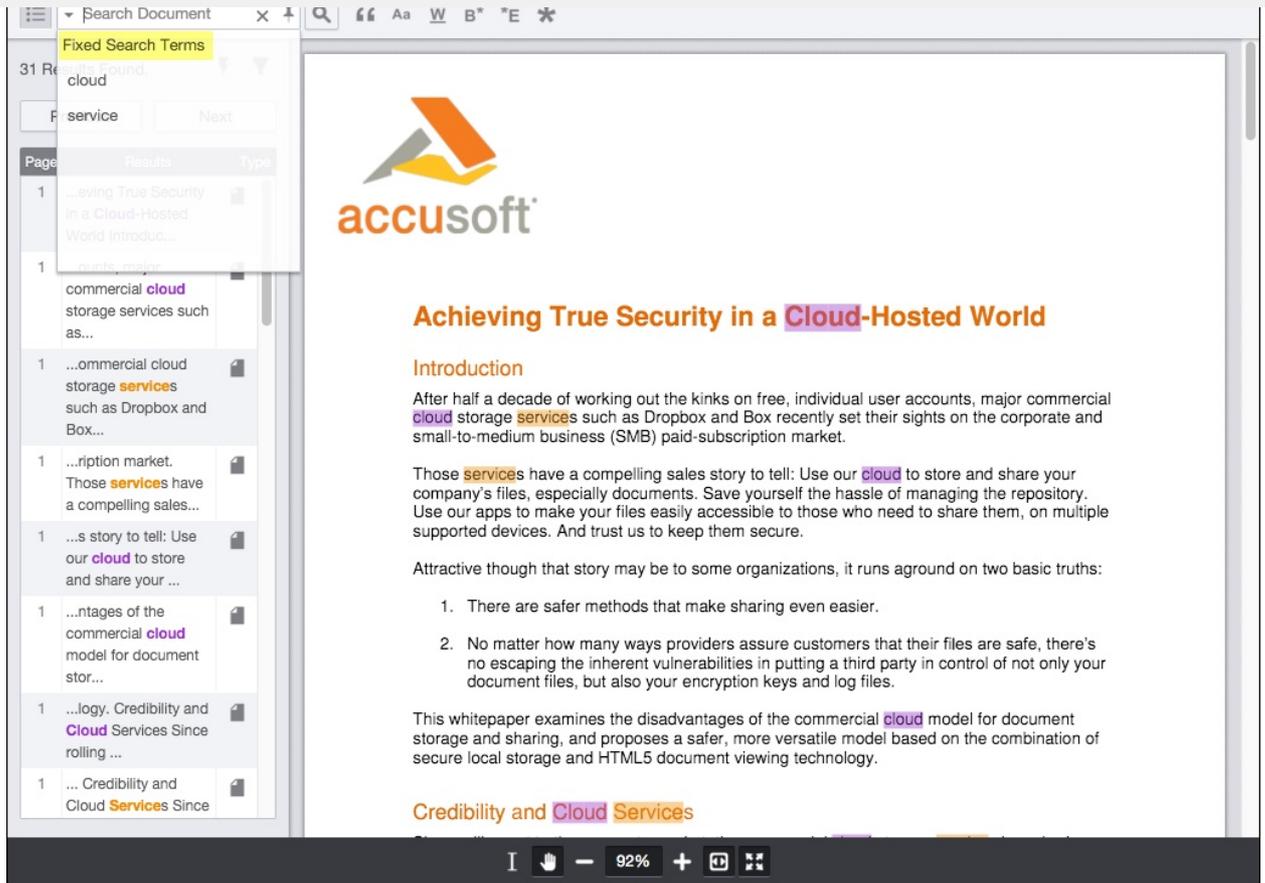
## Use the Fixed Search Terms Feature

To use the Fixed Search Terms feature:

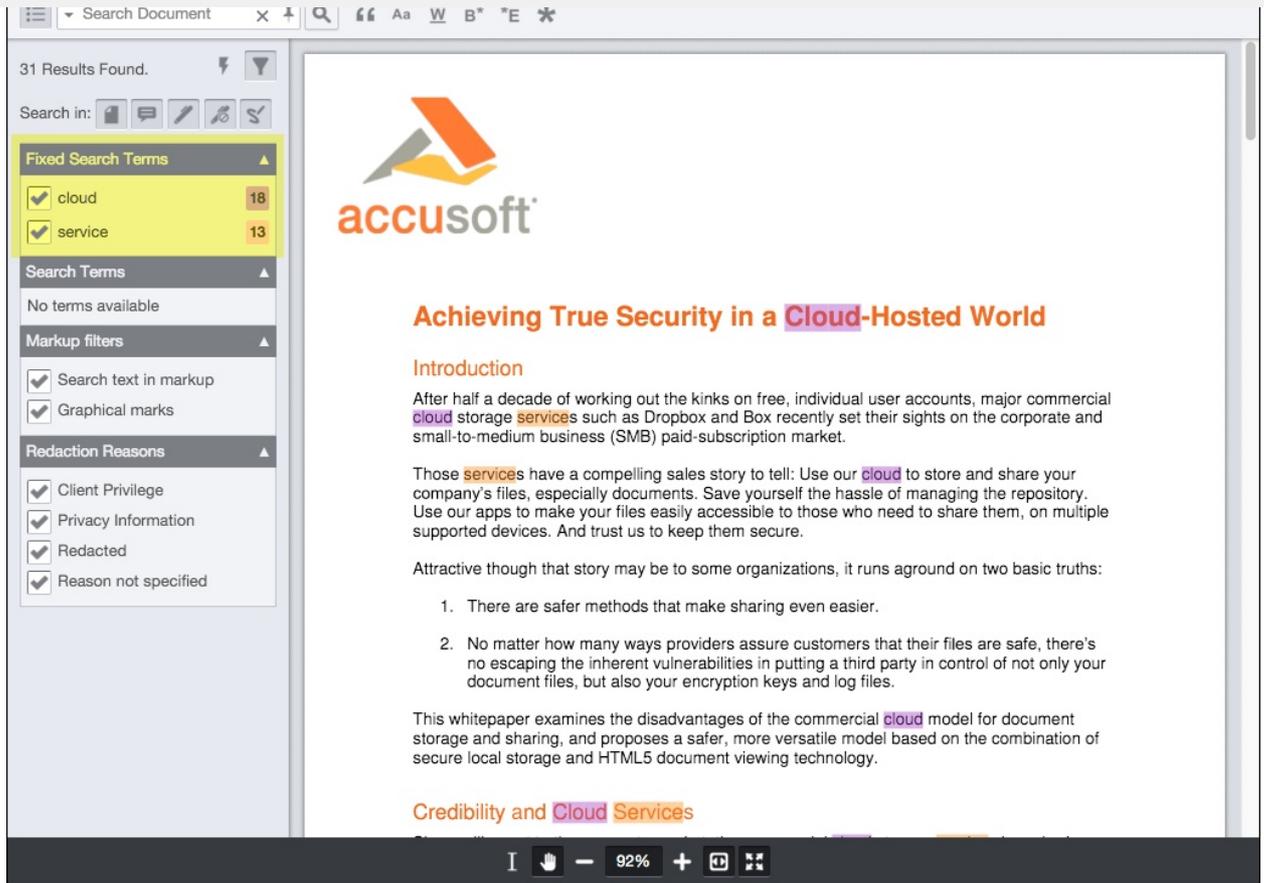
1. If a preloaded, fixed search is run when the document is loaded, the Viewer will open on the Search tab. A push-pin icon indicates that the Fixed Search is loaded and is displayed to the right of the Clear button in the Search input box:



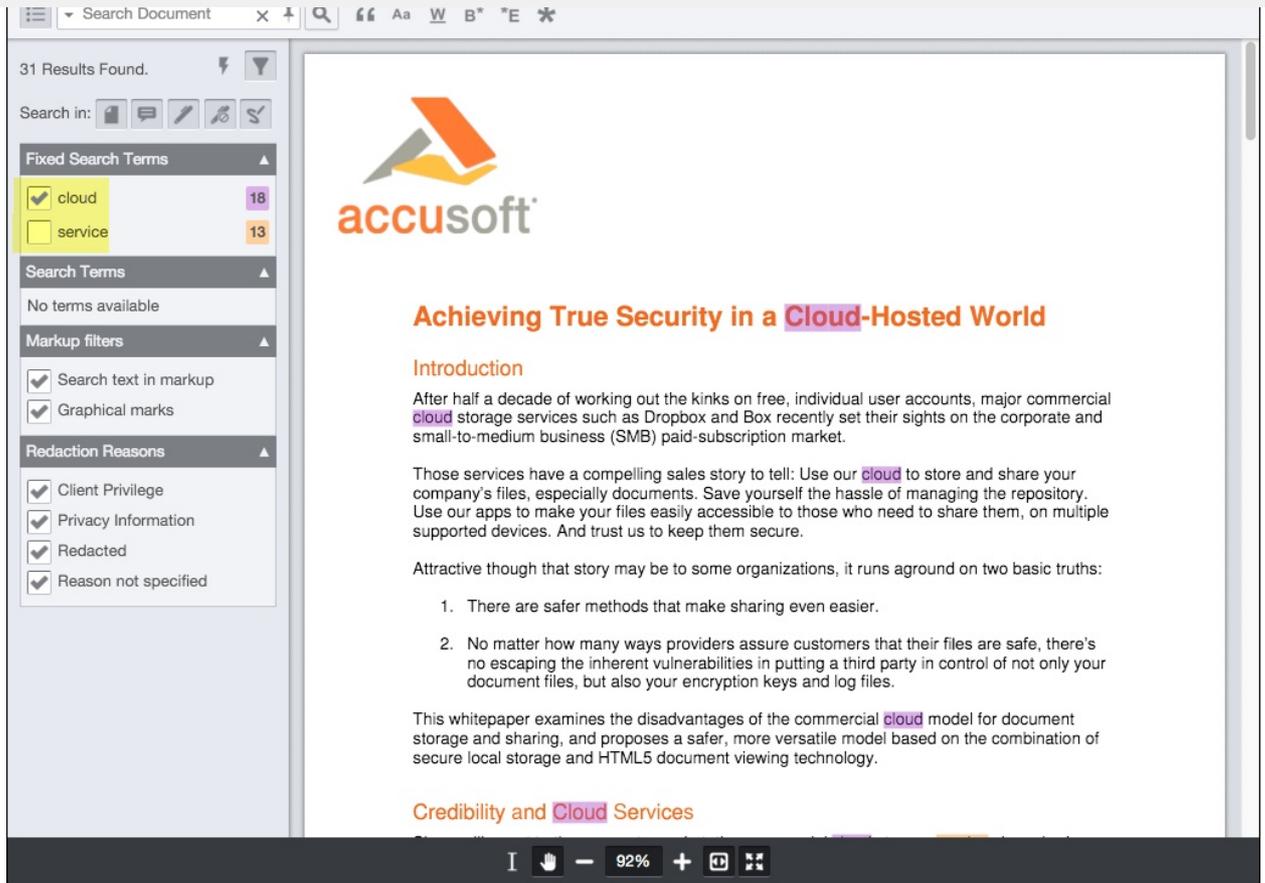
2. Click on the **push-pin** icon (in the Search box) and a list displays showing the Fixed Search Terms that are loaded in the document:



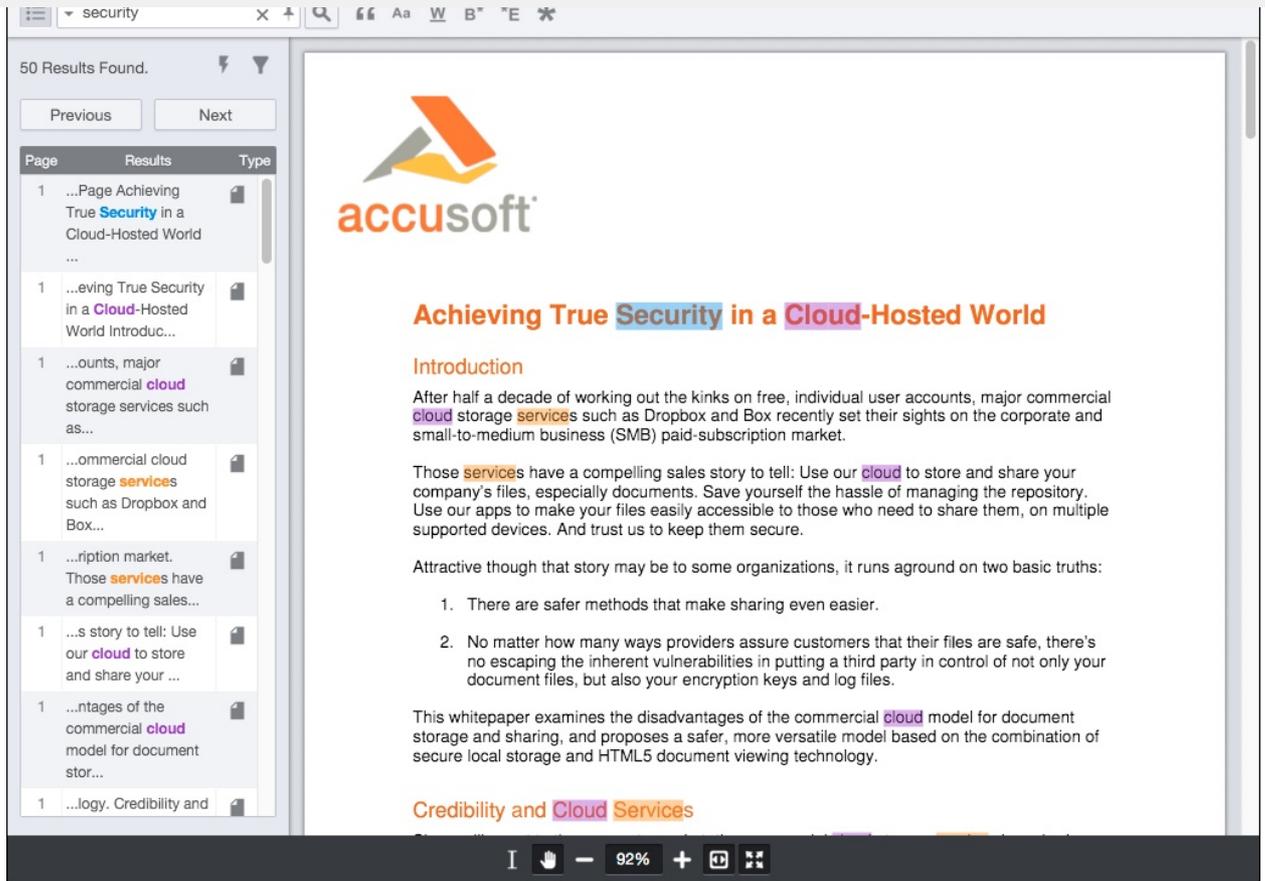
3. Click on the **Search** filter and you will see the Fixed Search Terms displayed at the top of the Search panel in their own section:



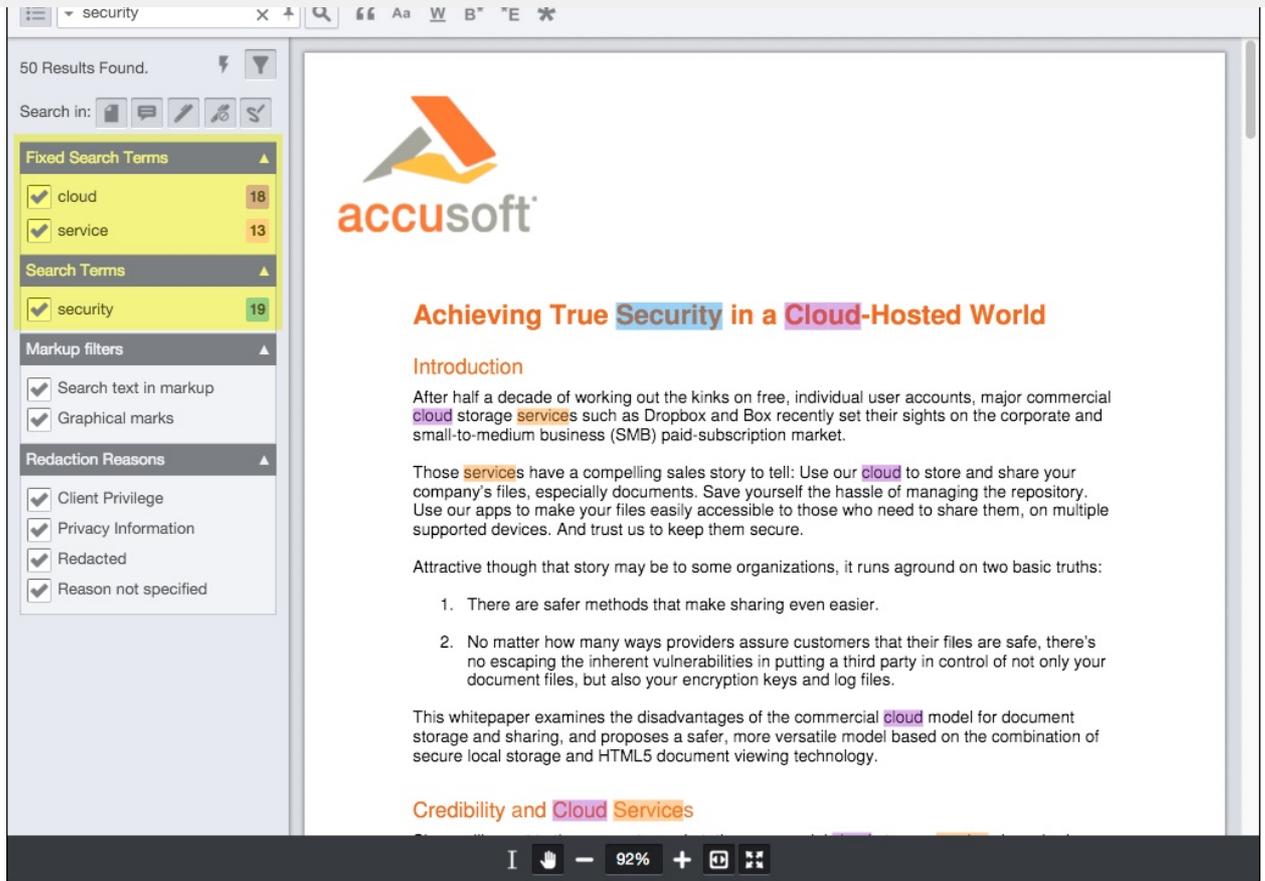
4. You can filter out a Fixed Search Term by de-selecting the **checkbox** next to the term:



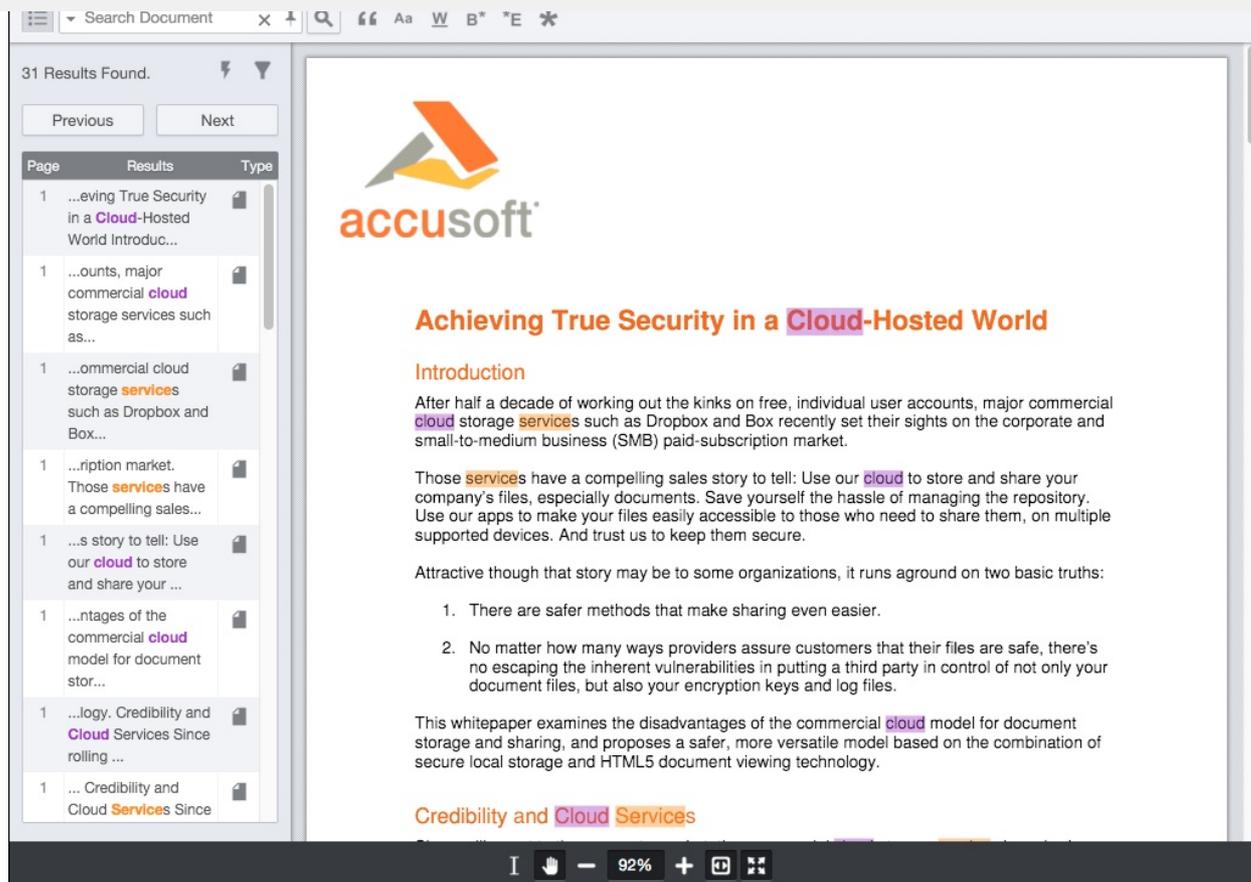
5. You can enter **additional search terms** along with the Fixed Search Terms:



6. If you enter additional search terms, they will display in the Search panel under the Fixed Search Terms:



- 7. To clear the search results, click on the **Clear** button. The additional search terms you entered will be cleared, the filters return to the default state and the Fixed Search Terms remain:



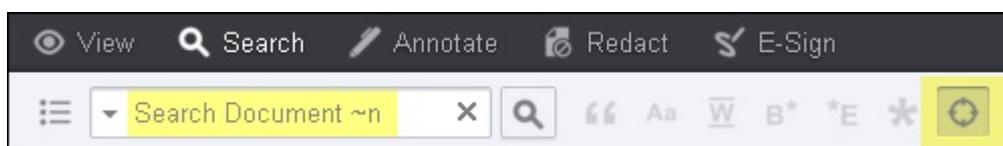
## Use the Proximity Search Feature

The Search tab contains the Proximity Search feature that helps you search for two terms within a specified distance of each other. You can perform a proximity search by using the ~n syntax in the search bar, where "n" represents how many words can be between the first and second search terms and still return a result. When the search results are displayed, they will be highlighted on the document for ease of viewing.

Please note the following:

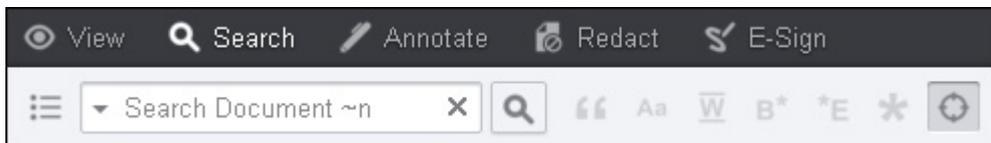
- When you select the Proximity Search icon, the other search options will be disabled.
- You must use the "~n" syntax; if you use anything else, you will get an error message.
- If you enter the correct "~n" syntax, but don't select the Proximity Search icon, a normal search is performed.

The following example shows the Search tab with the Proximity Search icon selected. The Search text box is pre-filled with the term "Search Document ~n" to help you remember to enter the terms with the ~n syntax:



To use the Proximity Search feature to search for terms:

1. From the **Search** tab, click on the **Proximity Search** icon:



2. In the **Search Document** text box, enter the **two terms** you want to search for along with the number (~n) to indicate the distance between the two terms and press **Enter**:



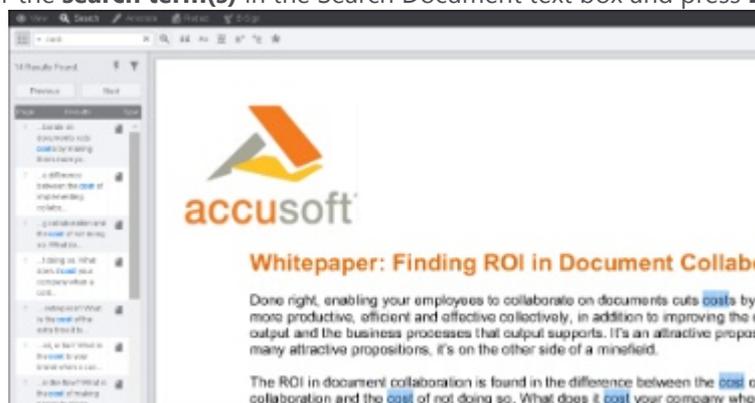
In the example above, the terms "collaborate documents ~3" were entered and searched for in the document. The terms are highlighted in the main viewing window and the results are also shown in the left-hand viewing pane for easier navigation with multiple search results. Note that the order of the terms you enter does not matter; you can enter "collaborate documents ~3" or "documents collaborate ~3" and the same results are returned for viewing.

## Redact Search Results

Use the following steps to search for specific terms and redact them throughout the entire document.

**To Redact Search Results:**

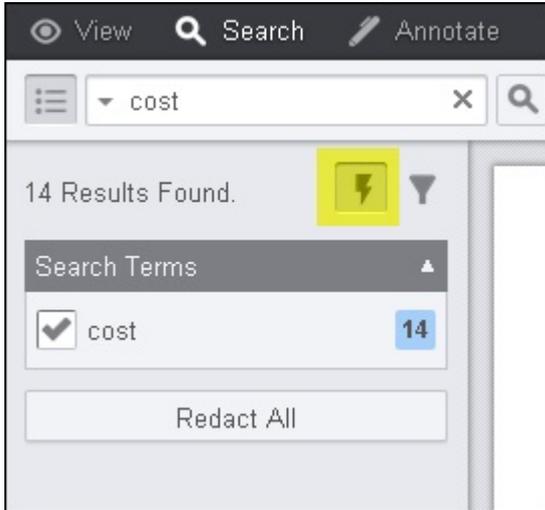
1. From the Search menu, enter the **search term(s)** in the Search Document text box and press **Enter**



(or click on the Search icon):

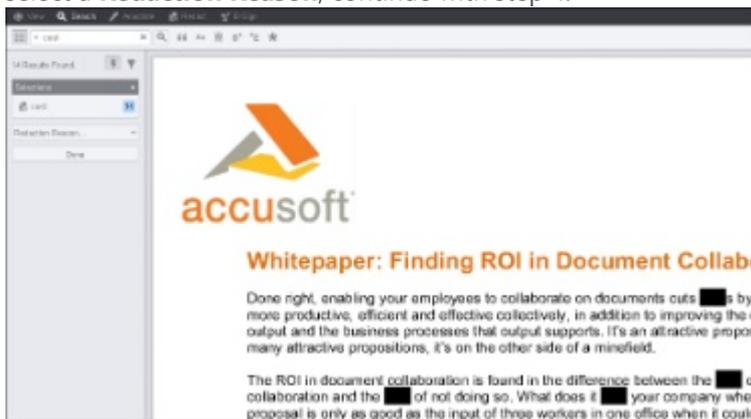
The Search Results pane is displayed with the number of results found.

the number of terms found in the document.

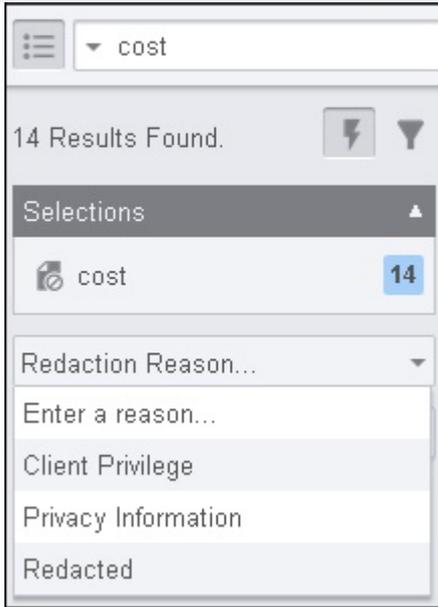


By default the checkbox for the terms is selected.

3. Click **Redact All** and the search results in the document are redacted. You can click **Done** or to select a **Redaction Reason**, continue with step 4.

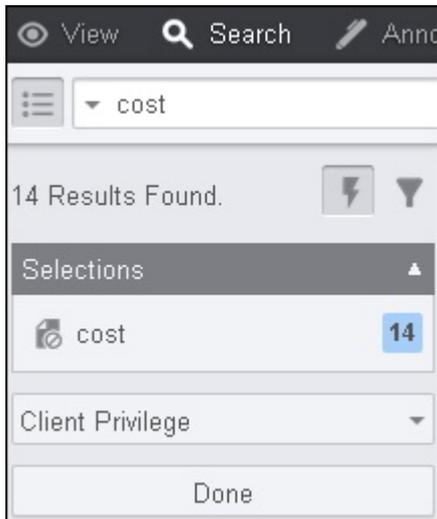


4. To select a Redaction Reason, click on the **drop-down** menu:



You can enter your own Redaction Reason or select from a pre-filled list of reasons.

5. When you are finished, click **Done**.



## Annotate Documents

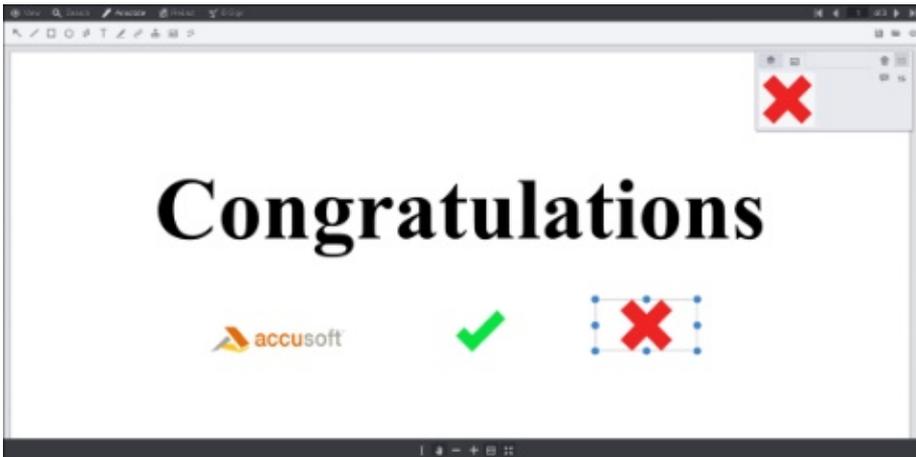
This section explains how to add, modify, and delete annotations, and include in burned documents:

- [Create Image Stamp Annotations](#)
- [Create a Polyline Annotation](#)
- [Create a Strikethrough Annotation](#)
- [Create a Text Hyperlink Annotation](#)
- [Use Annotation Layers](#)

- Load Annotations
- Save Annotations

## Create Image Stamp Annotations

Image Stamps are available under the Annotate menu. You can select a custom image and place it anywhere on your document. The following example shows Image Stamp Annotations displayed in the Viewer:



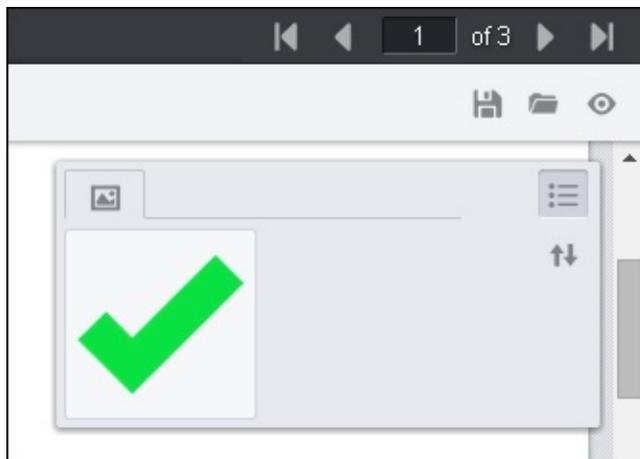
### To Add an Image Stamp Annotation to Your Document

Image stamp file size needs to be appropriate for the end users web browser and network capabilities. Also, IE8 can handle a maximum image stamp file size of 32KB.

1. Select the **Image Stamp** annotation:



The Image Stamp annotation context menu displays on the right-hand side of the Viewer:



2. Draw the **image stamp** on the document. The floating Context menu displays next to the image stamp:

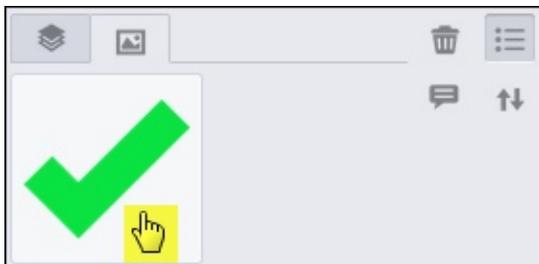


You can **Add Comment**, **Select** the image stamp, or **Cancel**. You can also select the image and move it anywhere you want on the document.

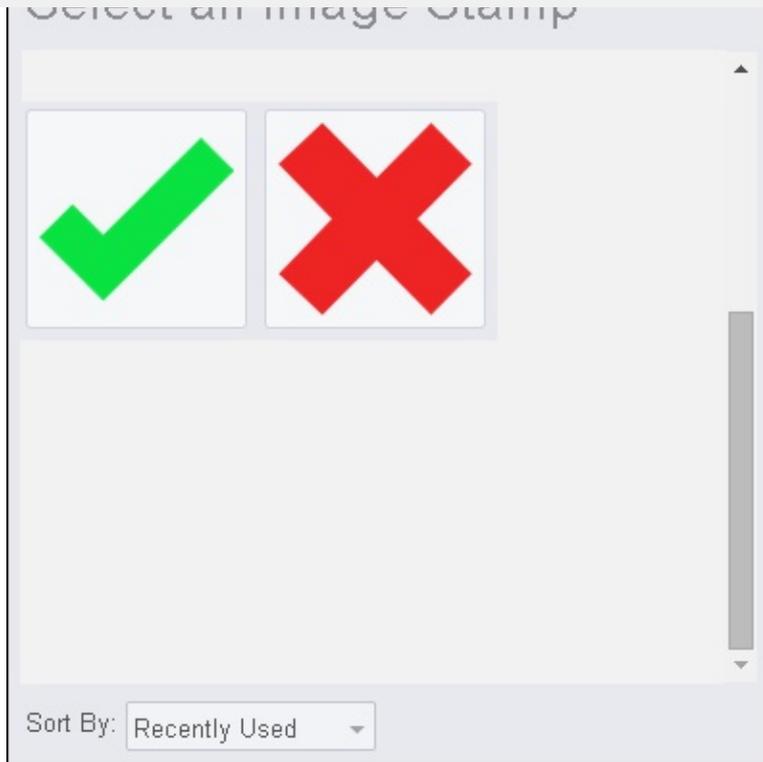
3. From the Context menu, you can **adjust the layer placement**, **add a comment** or **delete the image**:



4. To choose a different image stamp, click on the **image** in the Context menu:



5. The Select an Image Stamp dialog box displays:



6. Click on the **image** you want to use and the new image is placed on the document.

## Create a Polyline Annotation

You can add a Polyline annotation to your document. The Polyline is available from the Annotation menu:

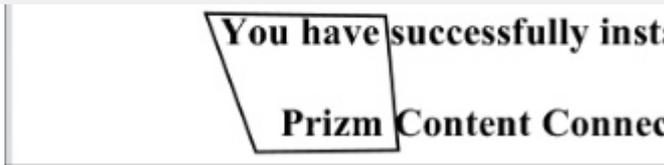


**To add a Polyline annotation to your document:**

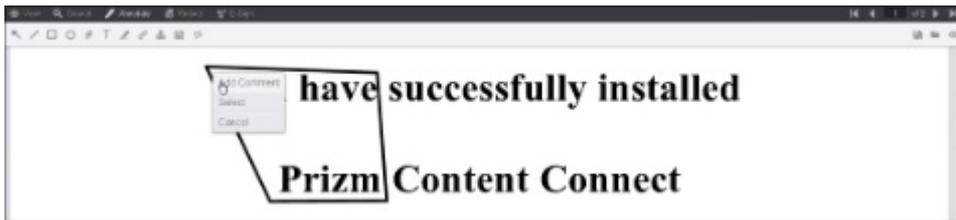
1. Click on the **Polyline** annotation:



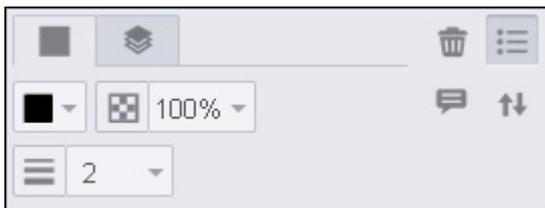
2. Draw the **Polyline annotation** where you want it to be on the document. Click to stop the line, then start drawing again. Double-click to complete drawing the Polyline annotation:



3. After you complete drawing the Polyline annotation, a context menu displays where you can **select to add a comment, select the annotation** or **cancel**:



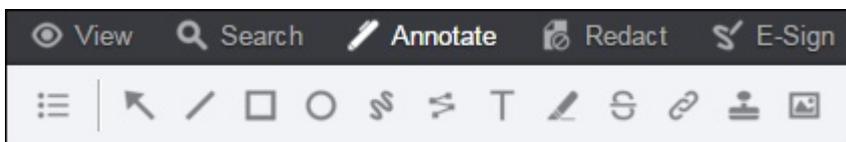
4. When you select the **Polyline annotation**, you can make other selections from the Comments menu, such as **line thickness, color, layer placement**, or **add a comment**:



5. To **delete** the Polyline annotation, click on the **Trash icon**.

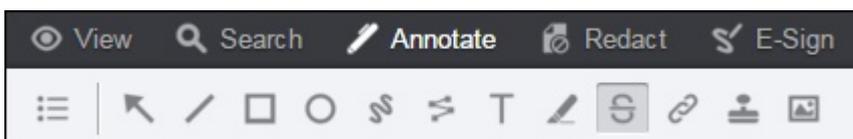
## Create a Strikethrough Annotation

You can add a Strikethrough annotation to your document. The Strikethrough is available from the Annotation menu:



To add a Strikethrough annotation to your document:

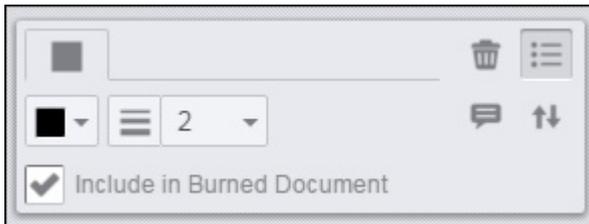
1. Click on the **Strikethrough** annotation:



2. Highlight the **text** that you want to strikethrough on the document:

...of three workers in one office when it could have benefited  
...open experts from across the enterprise? What is the cost  
...document when workers are collaborating inefficiently via  
...the cost to your brand when a customer or client sees the

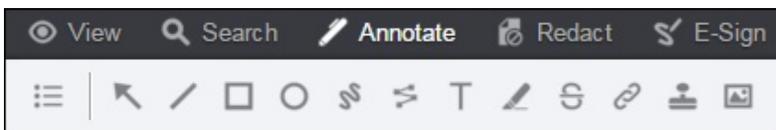
3. After you complete highlighting the text, a context menu displays where you can select to add a **comment**, select the **color** of the Strikethrough annotation, select the **size** of the Strikethrough, or click a **checkbox** to burn the annotation into the document:



4. To **delete** the Strikethrough annotation, click on the **Trash icon**.

## Create a Text Hyperlink Annotation

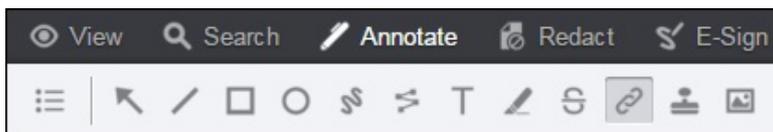
You can now add a Hyperlink annotation to your document. The Hyperlink annotation is available from the Annotation menu:



 You can easily modify your hyperlink annotation by using either touch or mouse.

**To add a Hyperlink annotation to your document:**

1. Click on the **Hyperlink** annotation:



2. Select the **text** that you want to add the hyperlink to and the link text box appears:



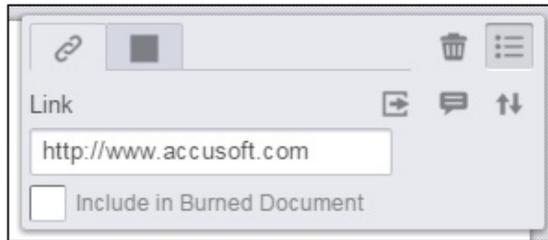
3. Type the **web address** into the link text box and select the checkbox:



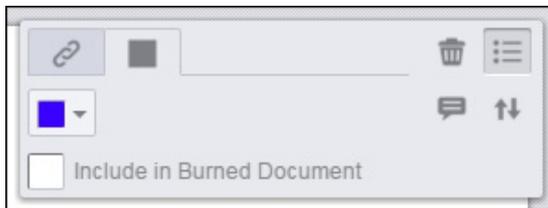
added to the link for you.



5. You can edit the hyperlink by clicking on the **pen icon**. From the comment menu, you can change the link, add a comment, select the checkbox to burn the hyperlink into the document or delete the hyperlink:



6. You can also select a different color for the hyperlink:



7. When finished, click on the **hyperlink** to go to the web site.

## Use Annotation Layers

This section explains how to load, save, and copy/merge annotation layers:

- [Work with Annotation Layers](#)
- [Load Annotations](#)
- [Save Annotations](#)

## Work with Annotation Layers

within a document and improves collaboration during document reviewing.

This section covers how to use annotation layers within a document and assumes you are already familiar with adding annotations and redactions to your document. If you have not added annotations or redactions to your document, refer to the following topics first: [Working with Annotations](#) and [Working with Redactions](#).

This topic contains the following sections:

- [Introduction](#)
- [How to Create & Save Your Own Layer under the 'My Annotations' Pane](#)
- [How to Load Annotations from other Reviewers from the 'My Annotations' Pane](#)
- [How to Edit the Name of an Annotation Layer under the 'My Annotations' Pane](#)
- [How to Hide an Annotation Layer under the 'My Annotations' Pane](#)
- [How to Load and Review Annotations from the 'Annotations for Review' Pane](#)
- [How to Show & Hide Annotations from the 'Annotations for Review' Pane](#)
- [How to Merge Annotations from the 'Annotations for Review' Pane](#)
- [How to Merge all Annotations from the 'Annotations for Review' Pane](#)

### Introduction

The Annotation Layer functionality is located under both the Annotate and Redact tabs:

Annotate tab:



Redact tab:

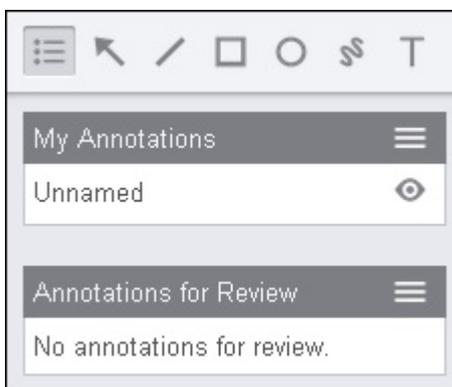


Once you click on the Annotation Layers icon, you can go to the other tabs (View, Search, E-Sign) and the Annotation Layers pane remains displayed.

To open the Annotation Layers pane, click the Annotation Layers icon from the toolbar:



There are two sections on the Annotation Layers pane that you can use - 'My Annotations' and 'Annotations for Review':



- My Annotations Pane - Load annotations, save annotations and edit the layer name.
- Annotations Review Pane - Load annotations and review, show/hide all annotations in one click, merge specific annotations or merge all annotations in one click.

## The 'My Annotations' Pane

### How to Create & Save Your Own Layer under the 'My Annotations' Pane

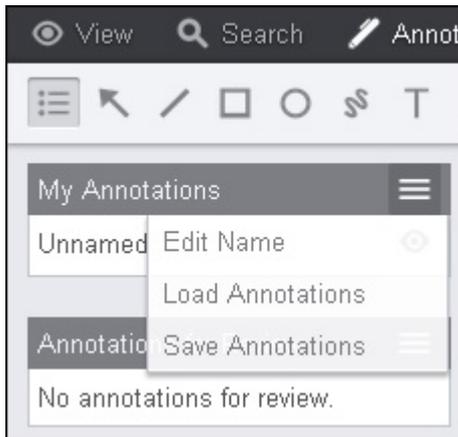
These steps assume you already know how to add annotations to a document. If you have not added annotations or redactions to your document before, refer to the following topics first: [Working with Annotations](#) and [Working with Redactions](#).

1. Draw an **annotation** on the document and add a **comment**.
2. Draw a **redaction** on the document and add a **comment**.

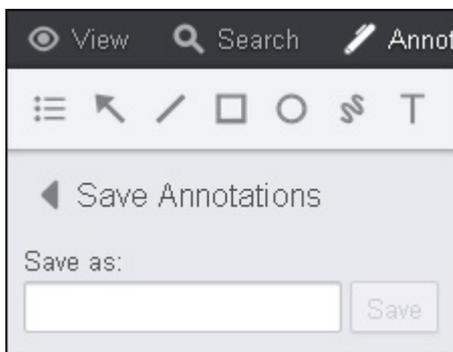
When you click on either the annotation or redaction, the comment displays a Reply field so another reviewer can respond to your comment in that layer:



3. Under **My Annotations**, click on the **Annotation Layers** icon and three options display (Edit



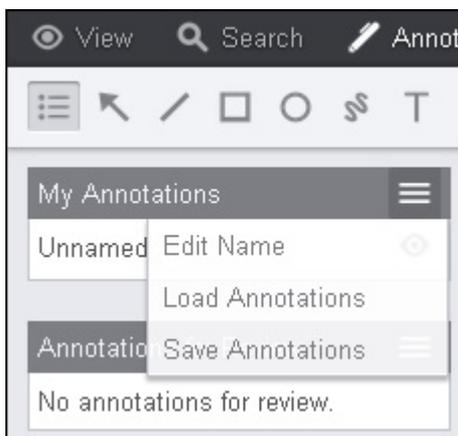
4. Select **Save Annotations**. The Save As field displays:



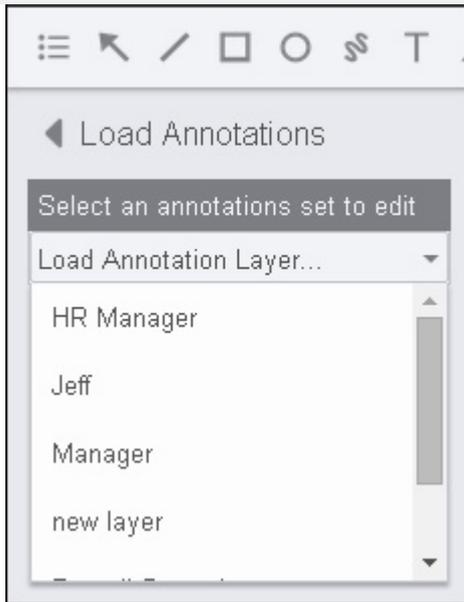
5. Enter the **Layer Name** and click **Save**.
6. A green notification bar displays stating the annotation and redaction are now saved in that layer.

### How to Load Annotations from other Reviewers from the 'My Annotations' Pane

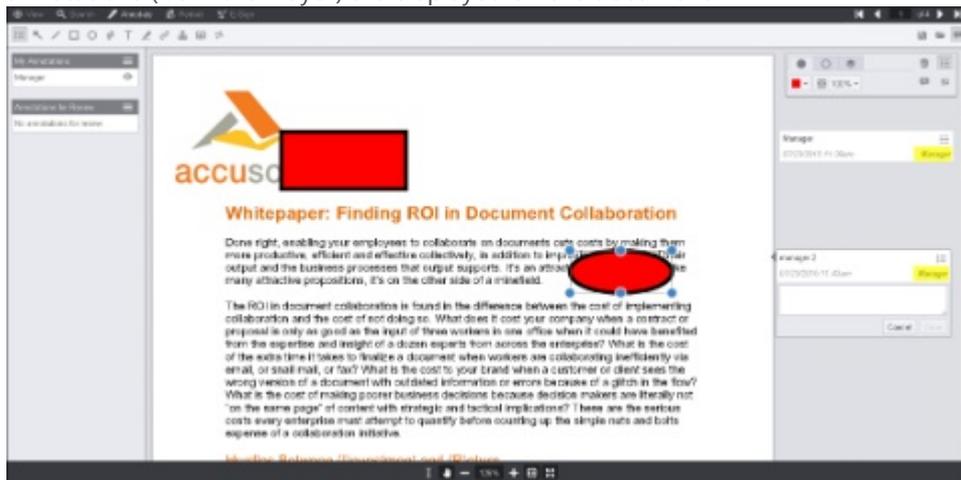
1. Under **My Annotations**, click on the **Annotation Layers** icon and three options display (Edit Name, Load Annotations, Save Annotations):



2. Select **Load Annotations**. The Load Annotations pane displays:

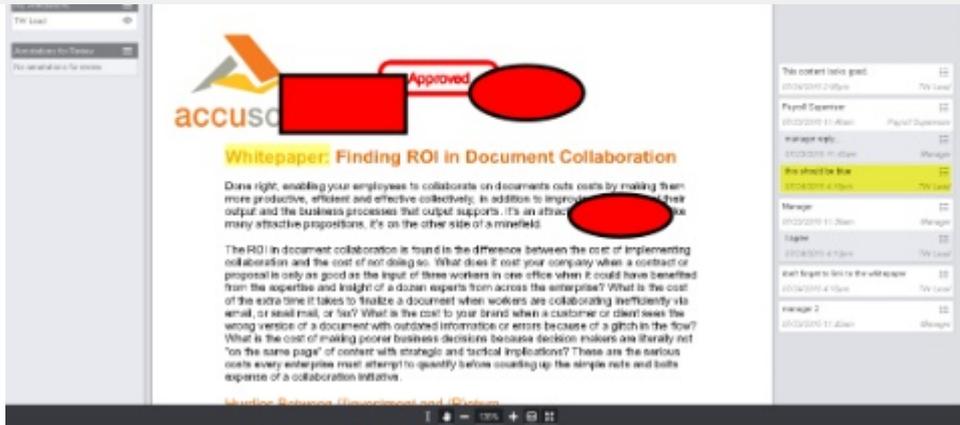


3. Select the **annotations set** you want to edit from the drop-down and another reviewer's annotations (from their layer) are displayed on the document:

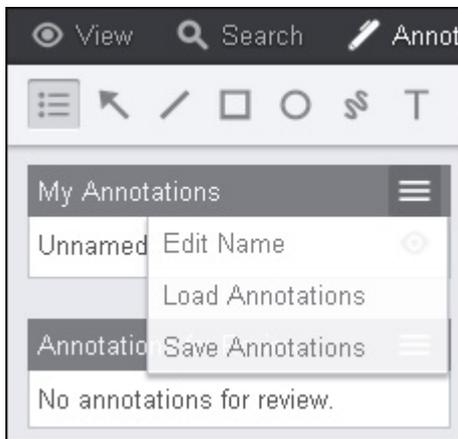


 The name of the layer that a comment belongs to is displayed next to the date and time in the comment (as shown above in the yellow highlighted area).

4. You can review their annotations and reply to their comments. You can also add your own annotations and comments to their layer:



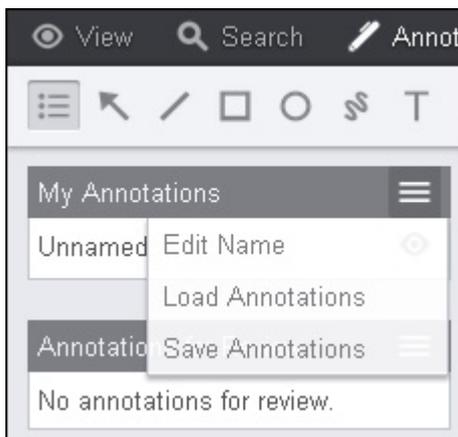
5. To save your changes, click on the **Annotation Layers** icon and select **Save Annotations**:



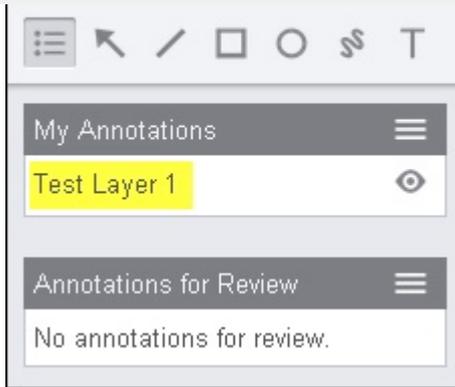
6. A green notification bar displays notifying you that your changes are now saved in that layer.

## How to Edit the Name of an Annotation Layer under the 'My Annotations' Pane

1. Under **My Annotations**, click on the **Annotation Layers** icon and select **Edit Name**:



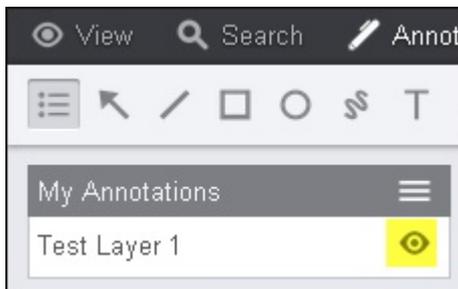
2. Add a **Layer Name** in the field and then press **Enter**:



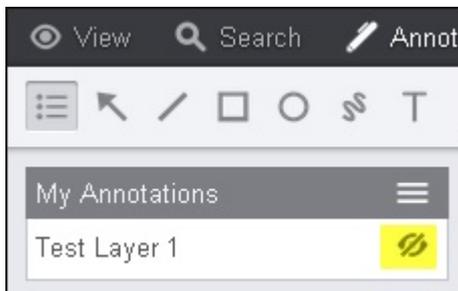
### How to Hide an Annotation Layer under the 'My Annotations' Pane

1. Under **My Annotations**, click on the **Eye** icon to toggle hide/show annotation layers on or off:

Annotation layers showing:



Annotation layers hidden:

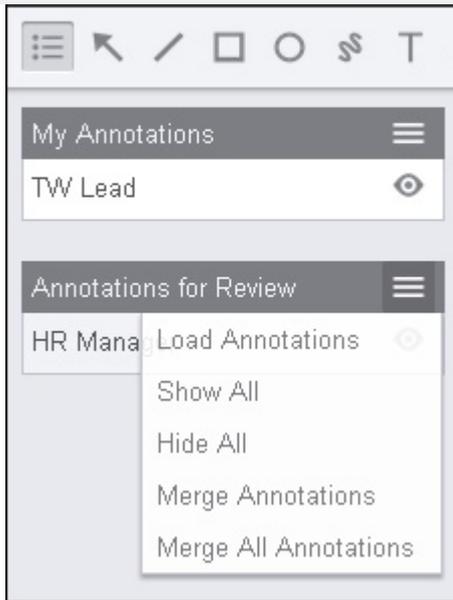


 If you hide a layer that contains redactions, those redactions will not burn with the document.

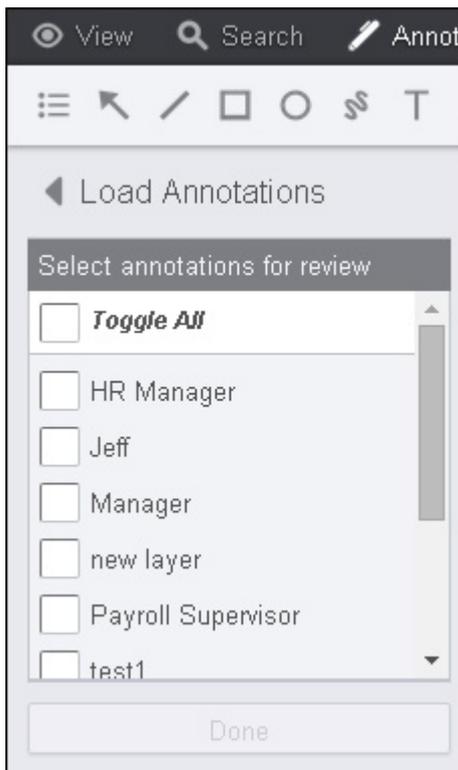
### The 'Annotations for Review' Pane

#### How to Load and Review Annotations from the 'Annotations for Review' Pane

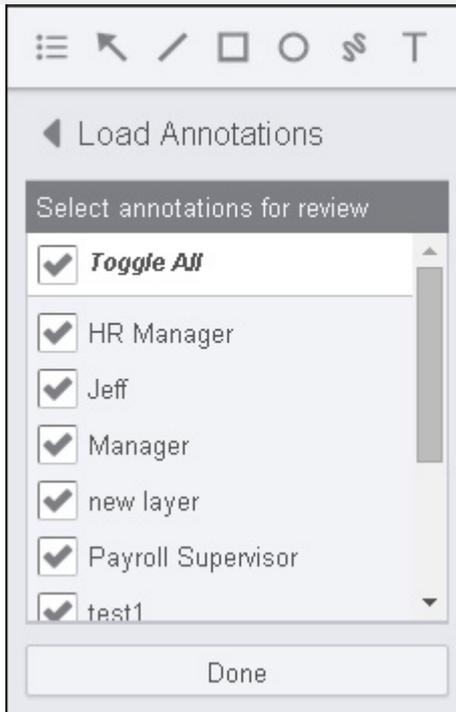
1. Under **Annotations for Review**, click on the **Annotation Layers** icon and five options display (Load Annotations, Show All, Hide All, Merge Annotations, and Merge All Annotations):



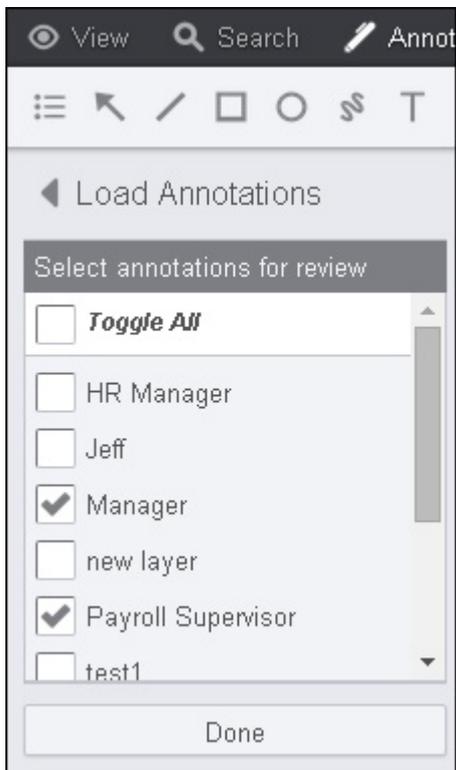
2. Click on **Load Annotations** and a list of annotations for review are displayed:



3. If you want to select **all the annotations** for review, select **Toggle All** and click **Done**:



Or if you only want to load and review specific annotations, select the **checkbox** in front of each set of annotations you want to review and click **Done**:

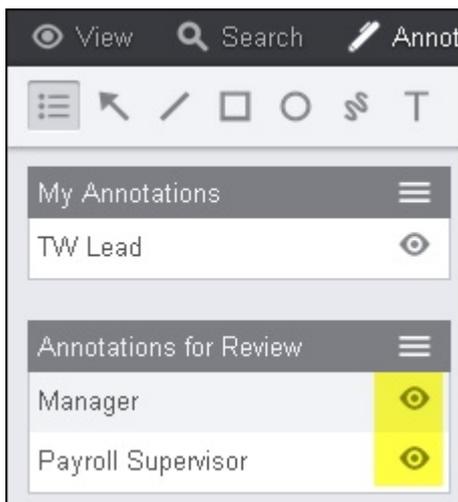


### How to Show & Hide Annotations from the 'Annotations for Review' Pane

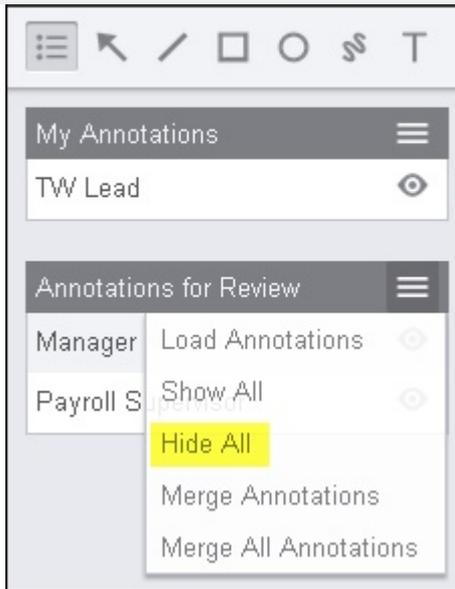
1. To show all annotations, click on the **Annotation Layers** icon and select **Show All**:



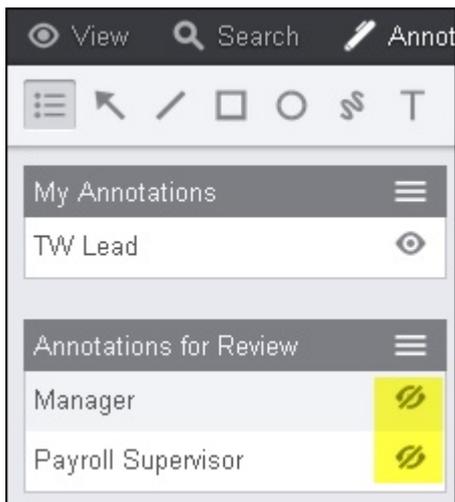
The Eye icon is displayed:



2. To hide all annotations, click on the **Annotation Layers** icon and select **Hide All**:



The Eye icon is displayed showing they are now hidden:

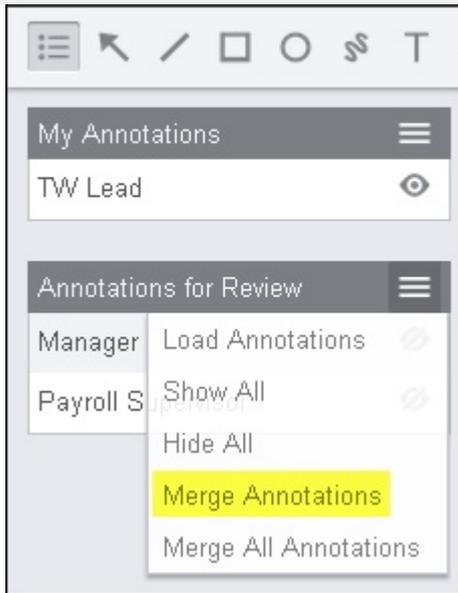


 If you hide a layer that contains redactions, those redactions will not burn with the document.

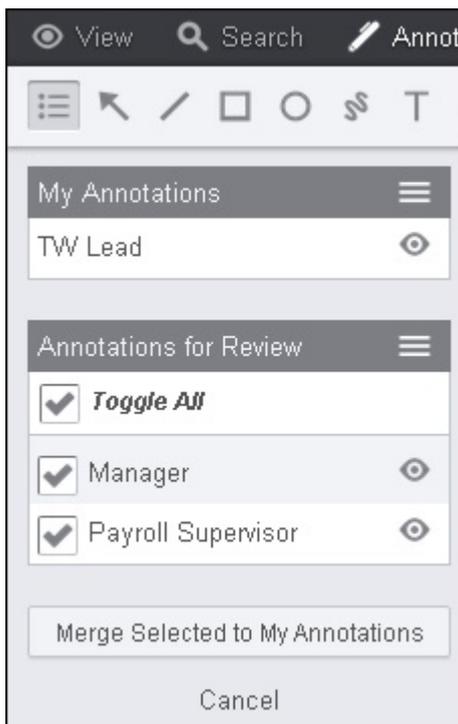
### How to Merge Annotations from the 'Annotations for Review' Pane

Note that when you merge annotations from one layer to another layer, the original layer name will stay with any comments made in the original layer.

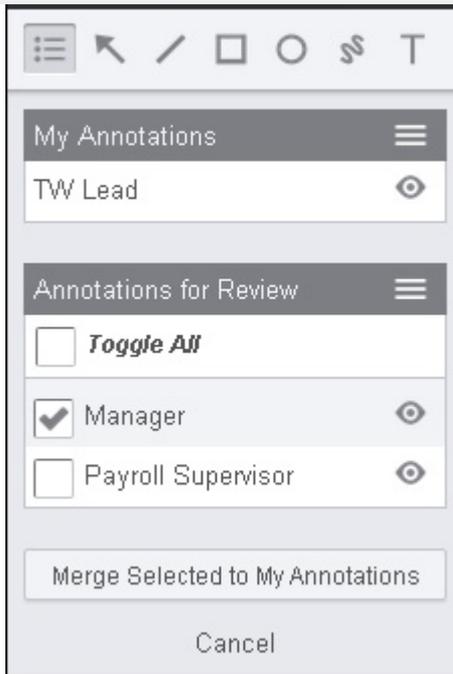
1. To merge annotations, click on the **Annotation Layers** icon and select **Merge Annotations**:



2. To select all annotations, select **Toggle All**:



If you want to select specific annotations, select the **checkbox** in front of the annotations you want to merge:

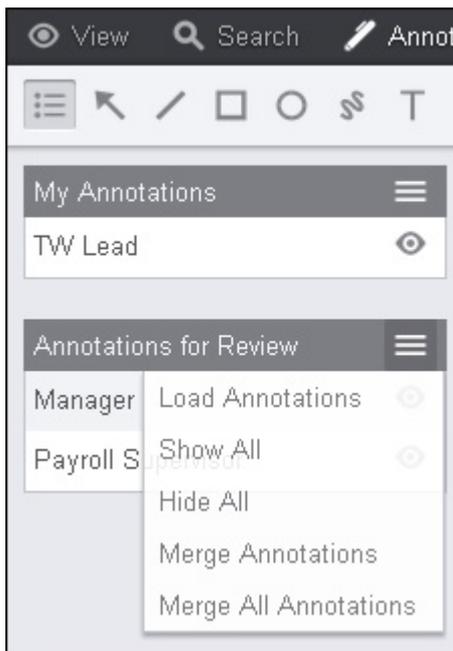


3. When you are done making your selections, click **Merge Selected to My Annotations**. The annotations that you selected have been merged with your annotations.

### How to Merge all Annotations from the 'Annotations for Review' Pane

Note that when you merge annotations from one layer to another layer, the original layer name will stay with any comments made in the original layer.

1. To merge all annotations, click on the **Annotation Layers** icon and select **Merge All Annotations**:

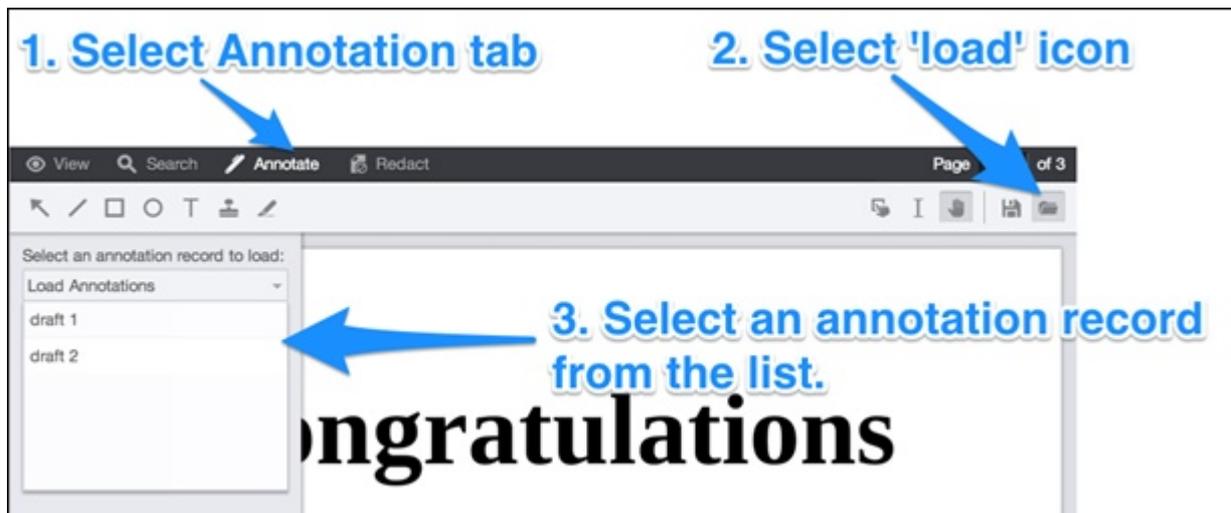


All annotations have been merged with your annotations.

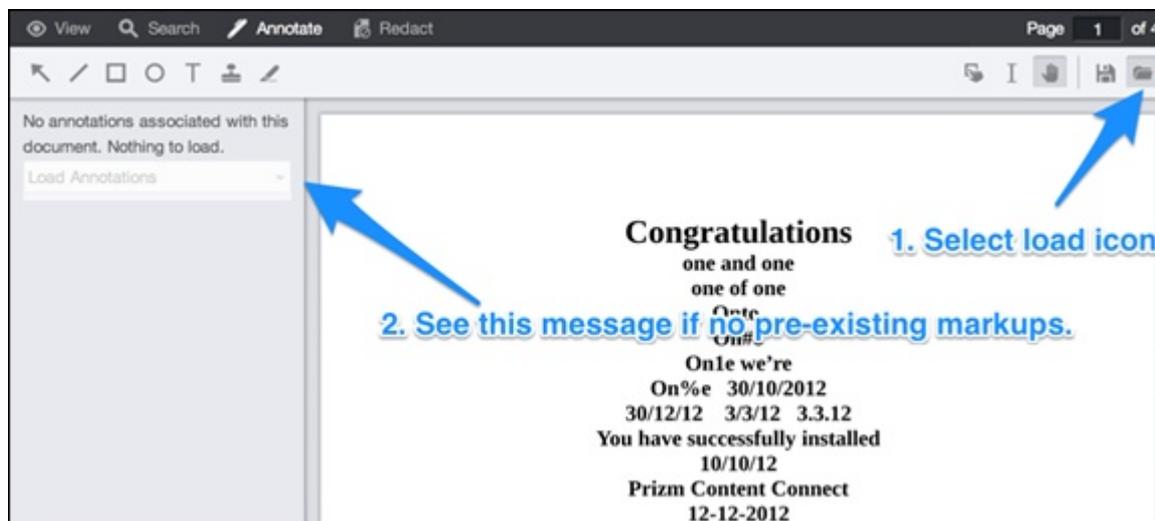
### LOAD ANNOTATIONS

#### Loading Previously Saved Annotations

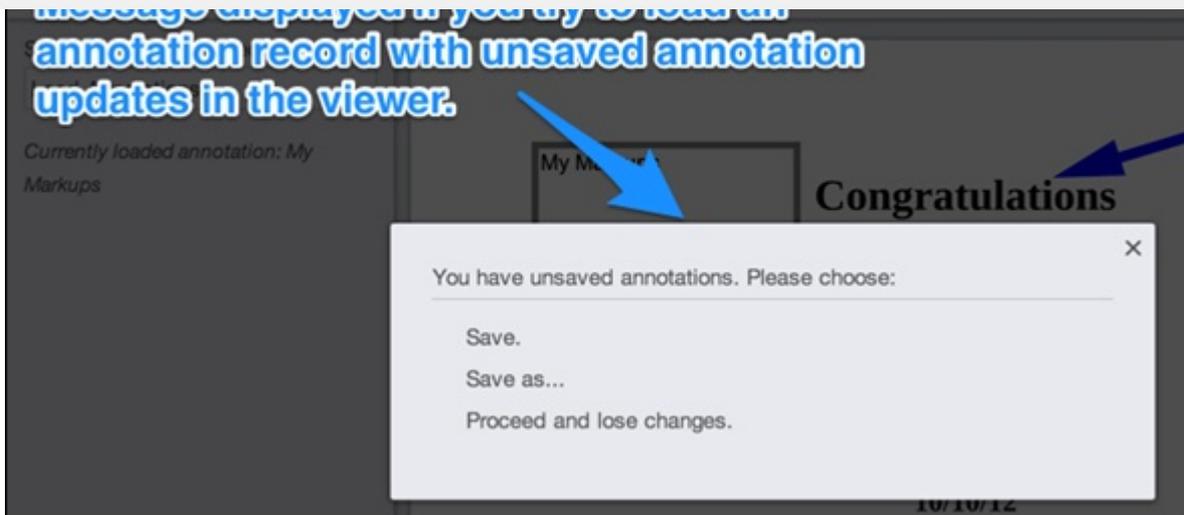
To load previously saved annotations, follow the steps shown below:



If no pre-existing annotation files exist for the loaded document, the message shown below appears. Also, the annotation file selection list will be disabled:



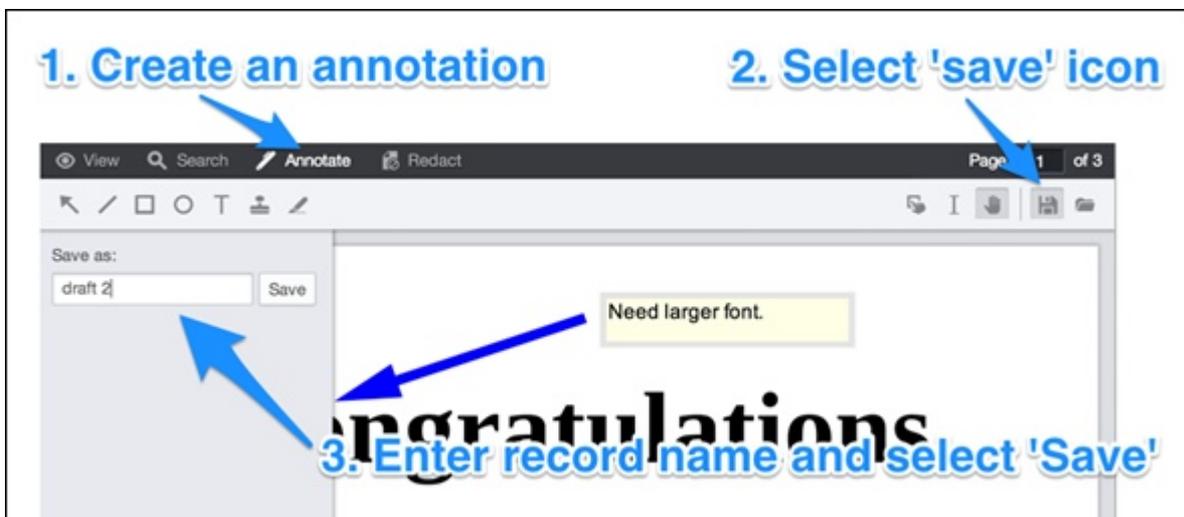
The **Unsaved** warning dialog below appears if you attempt to load an annotation file and might lose existing, unsaved annotations. To continue, select from one of the three available options:



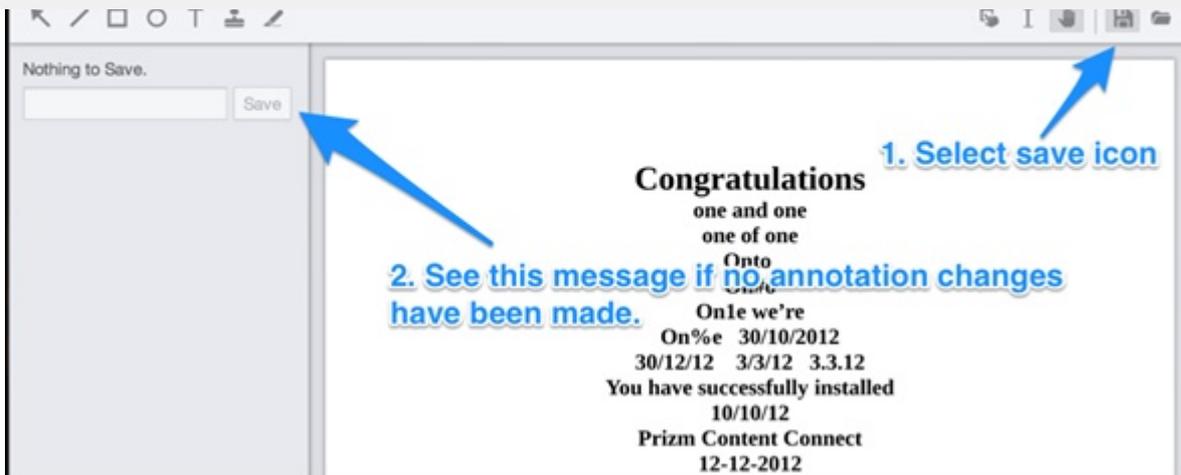
## Save Annotations

### Saving Annotations

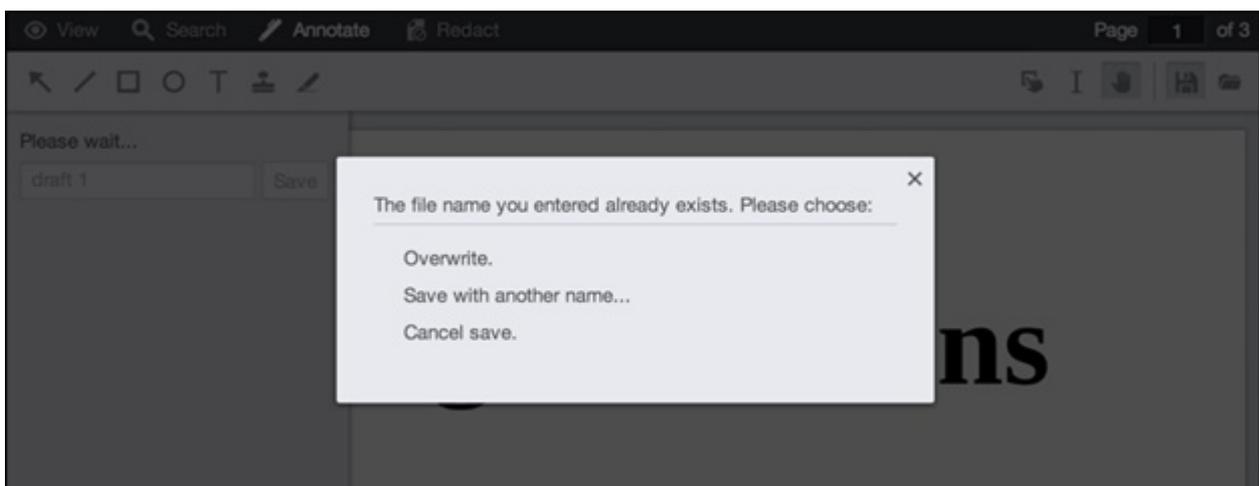
After adding annotations to a document, you can save the annotations for later use by following the steps shown below:



If no annotations have been added, modified, or deleted from the loaded document, the **Nothing to Save** message appears:



If you attempt to save annotations with a pre-existing file name, you will be prompted as shown below. To continue, select from one of the three available options:



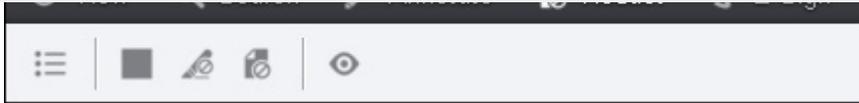
## Redact Documents

This section explains how to add, modify, and delete redactions, use redaction reasons and the redaction view mode:

- [Create a Full Page Redaction](#)
- [Use Redaction Reasons](#)

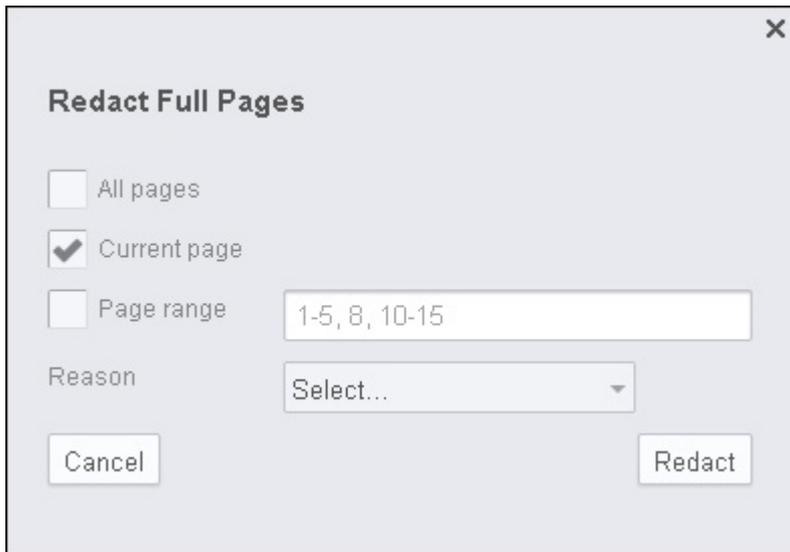
## Create a Full Page Redaction

You can create a full page redaction in your document. You have the option to redact the current page, all pages or a range of pages and you can select a Redaction Reason to apply to the pages that will be redacted. The Redact Full Pages mouse tool is available from the Redact menu:

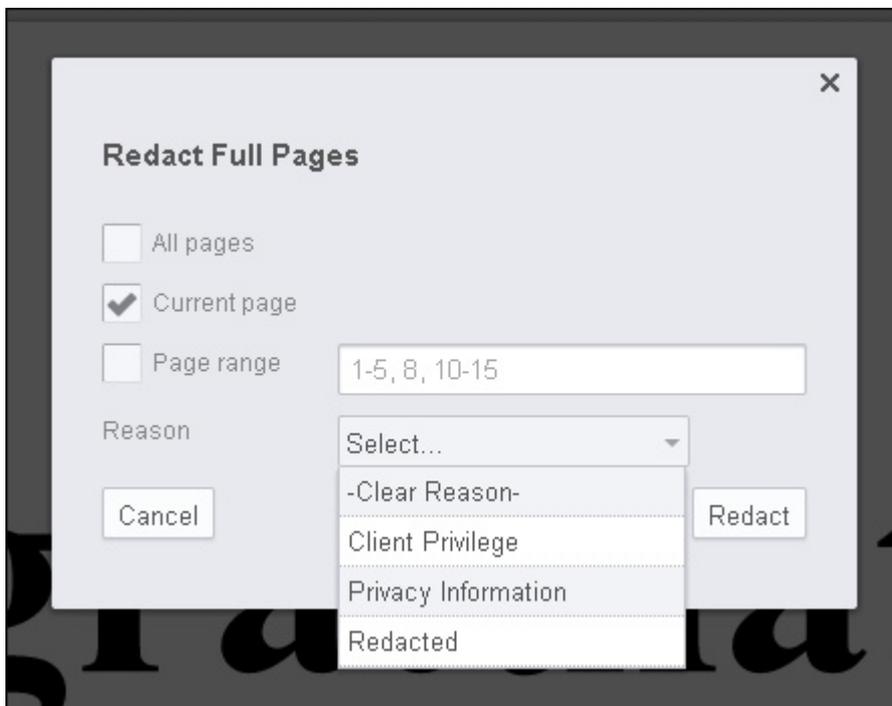


To create a full page redaction:

1. Click on **Redact Full Pages** on the Redact menu. The Redact Full Pages dialog box displays:



2. Select either **Current Page** (default), **All Pages** or **Page Range**.
3. If you want to apply a Redaction Reason, click on the **drop-down** and select **the reason**:

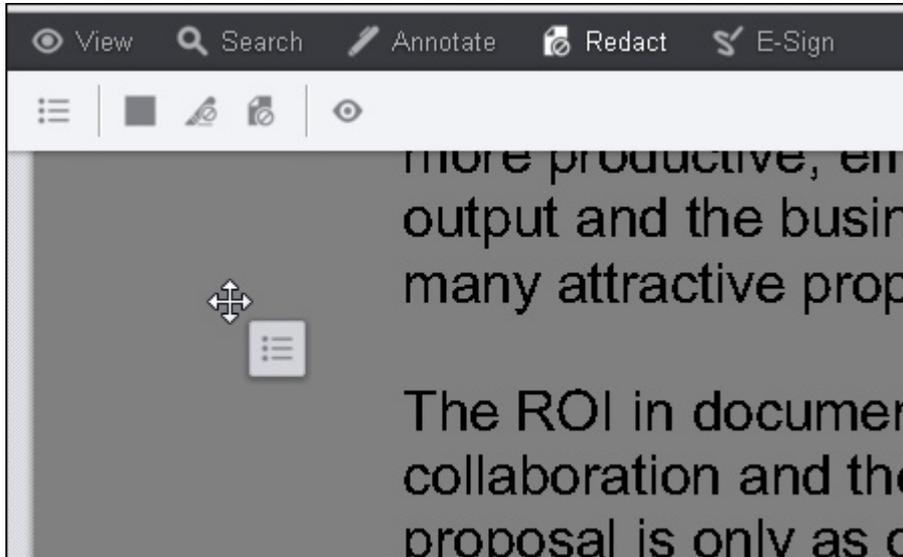


4. When you have finished making your selection, click **Redact**. The Full Page Redaction is applied to your document.

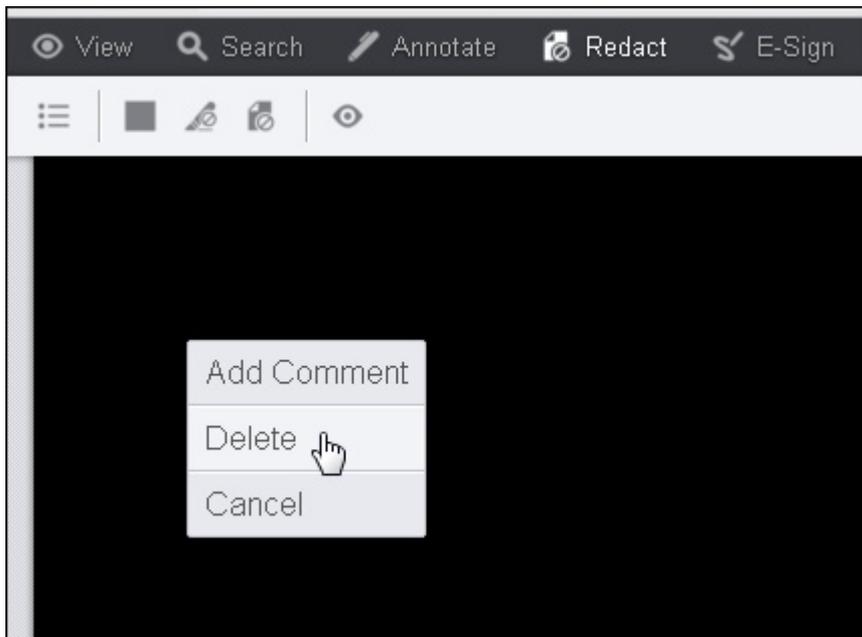
 You can print and review redacted text in a document by selecting the 'Review Redacted Text'

### To delete a full page redaction:

1. Click on the page that has the full page redaction. An immediate action menu displays:



2. Click on the immediate action menu and select **Delete**:



 Note that you can only delete one full page redaction at a time. If you have more full page redactions in the document to delete, you must go to that page and repeat steps 1 and 2 above.

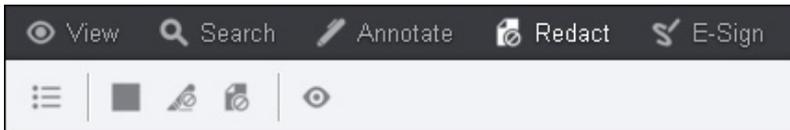
## Use Redaction Reasons

Redaction reasons allow redacted content to be replaced with a redaction reason that can be configured by the

What happens to the redaction reasons text:

- It gets burned into the document.
- It replaces the text in the burned in content.
- It is displayed in the Viewer UI.

Redaction reasons are located under the Redaction menu:

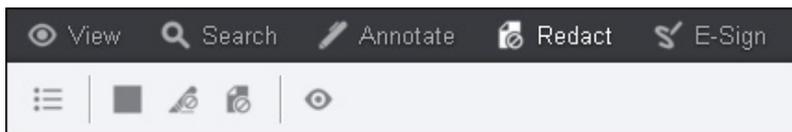


Redaction Reasons are not viewable when the Redaction View Mode is turned on.

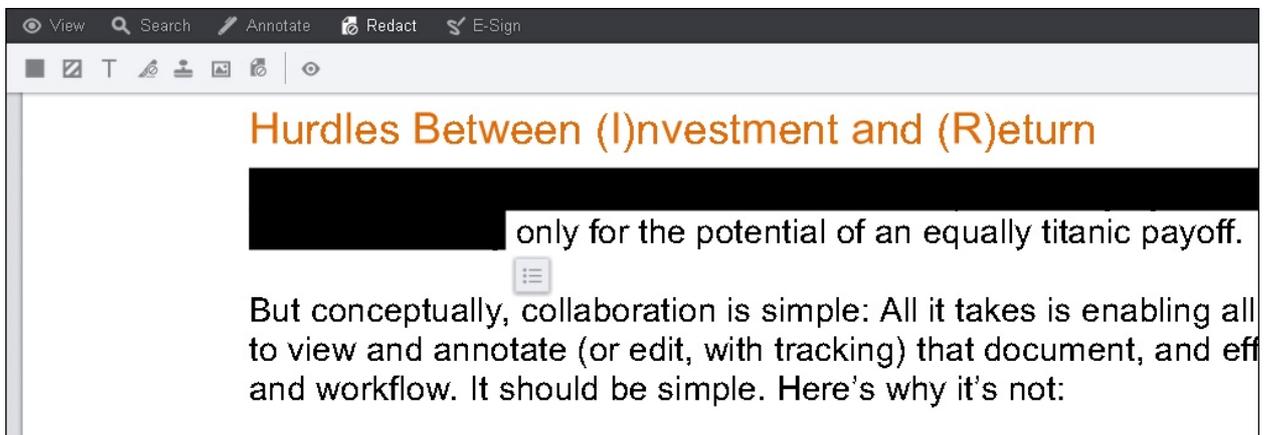
### To Add & Save a Redaction Reason to a Document

Redaction reasons are available with the **Filled Rectangle** and **Text Selection** redactions. The following instructions use the **Text Selection** redaction.

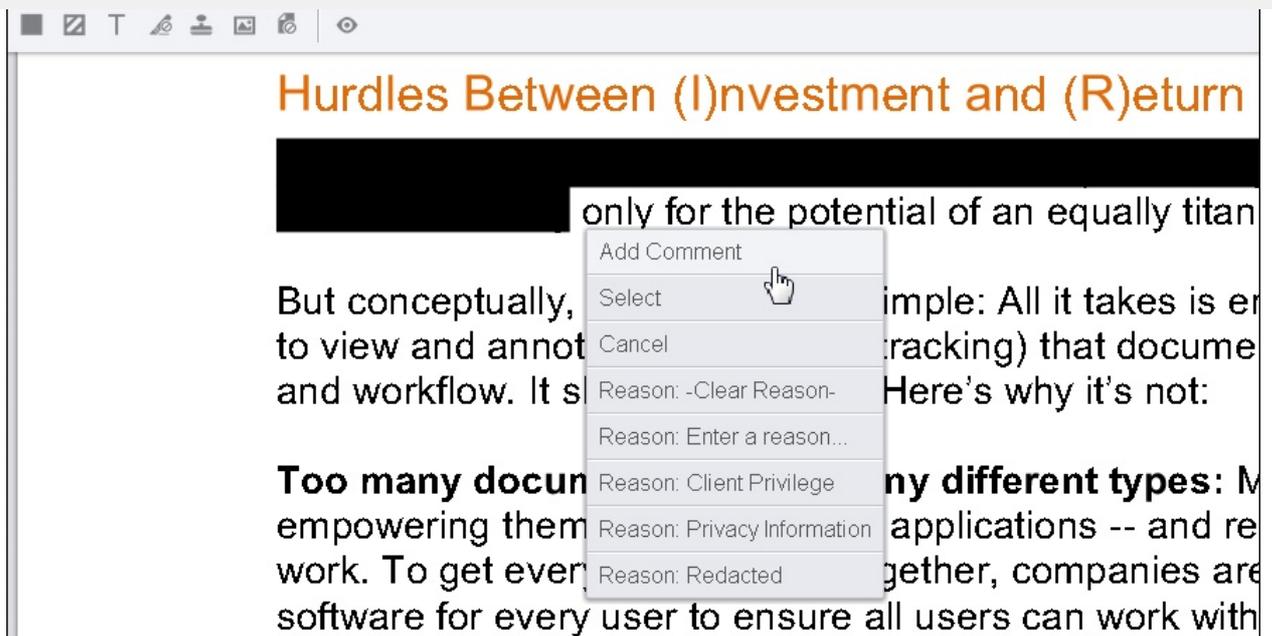
1. Click on the **Text Selection** redaction icon:



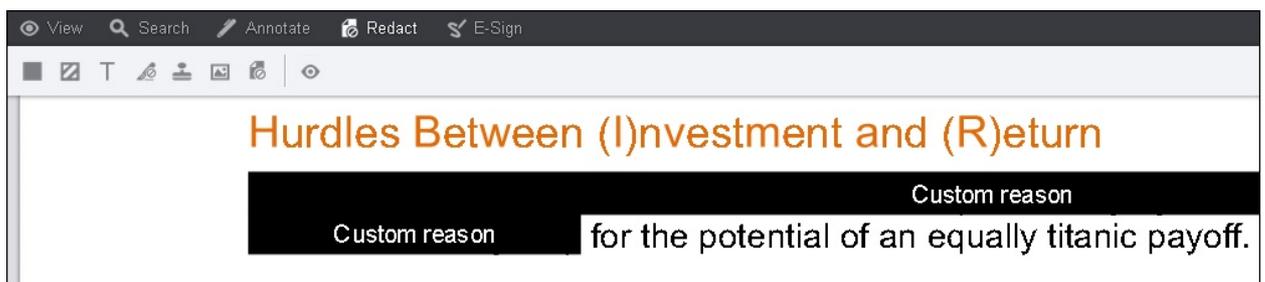
2. Select the **text** you want to redact and a context menu displays:



3. Hover over the **context menu** and the available options display:



4. Select the **Redaction Reason** you want to use or select **Enter a reason...** to create your own custom redaction reason:



If you hover over the redaction, it is transparent. Note that the redaction reason text is displayed in the redacted area. If there are multiple lines of redacted text, the reason is displayed on each line of text.

5. To edit the **Redaction Reason**, click on the **Redaction** and the Comments menu displays:

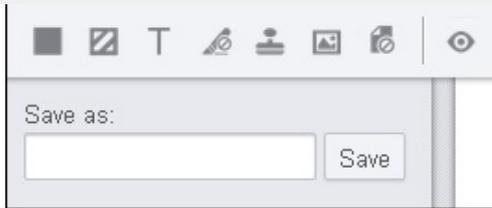


Click the **drop-down** to select another **Redaction Reason** or edit the **Custom Reason**. You can also add a comment or delete the Redaction from the Comments menu.

6. When you are ready to save your redactions and annotations, click the **Save** icon:



7. The **Save As** field displays on the left-hand side of the Viewer:



8. Enter a name for the redaction and click **Save**.

## How to Load Redaction Reasons

Redaction reason data is automatically loaded from a file named "redactionReason.json" located at the root of each sample provided. If the redaction reason functionality is not needed, then removing this file will disable the ability to add reasons to redactions in the Viewer.

## How to Set a Default Redaction Reason

**To set a default redaction reason:**

1. Open the **redactionReason.json** file.
2. Edit the **reasons property** so at most one item has a property called "defaultReason" set to the value of "true". If more than one item is set as default, then an error will be displayed when the Viewer loads.

The following example shows a reason set as the default in the redactionReason.json file:

### Example

```
...  
{"reason": "Content is removed from this section.", "defaultReason": true}  
...
```

## How to Automatically Apply Redaction Reasons

When the Viewer loads, the values from the redactionReason.json file will be read and viewer options will be appropriately set. If a "autoApplyDefaultReason" is set to true, then newly created rectangle or text selection redaction marks will automatically display the selected default reason text. If the "enableRedactionReasonSelection" is set to true then the reasons listed in the file will be made available as selectable options in the Viewer interface when creating or updating rectangle or text selection redaction marks. If the "enableRedactionReasonSelection" is set to false, then users will not be able to modify the default reason after it has been applied.

## JSON Input for Setting Up Redaction Reasons

Use the following example to set up redaction reasons:

### Example

```
{  
  "autoApplyDefaultReason": true,  
  "enableRedactionReasonSelection": true,  
  "enableFreeformRedactionReasons": true,  
  "maxLengthFreeformRedactionReasons": 40,  
  "reasons": [  
    {"reason": "Content is removed from this section.", "defaultReason": true}  
    , {"reason": "Sensitive data removed from the document."}  
    , {"reason": "Confidential."}  
  ]  
}
```

The following table shows the name, type and description available:

Name	Type	Description
autoApplyDefaultReason	boolean	When set to true, default redaction reason will be automatically applied when mark is created. If this property is not provided it will be considered as false.
enableRedactionReasonSelection	boolean	When set to true, the redaction reasons will be loaded into the Viewer and available for the end user to select the reason from the list. If this property is not provided it will be considered as true.
enableFreeformRedactionReasons	boolean	When set to true, users are no longer limited to the preset list of reasons, instead, they are able to type any redaction reason.
maxLengthFreeformRedactionReasons	int	When used with the "enableFreeformRedactionReasons" property, this parameter will limit the length of the reason that a user can type.

The following table shows how the options in the JSON file affect what tabs are displayed in the redaction reason context menu:

enableRedactionReasonSelection	autoApplyDefaultReason	Reason Dropdown Tab	Font Tab
true	true	displayed	displayed
true	false	displayed	displayed
false	true	hidden	displayed
false	false	hidden	hidden

## Use the Comment Feature

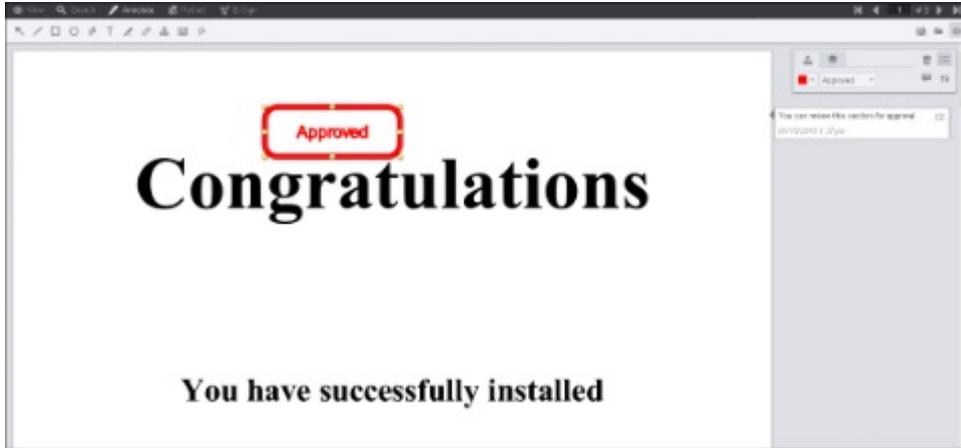
This section explains the purpose of the comment viewing modes, and how to add, edit, or delete comments on your own layer, and how to respond to comments on other layers:

- [Use Annotation Comments](#)
- [Use E-Signature Comments](#)
- [Use Redaction Comments](#)
- [Use Skinny Comments](#)

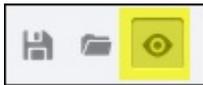
## Use Annotation Comments

Comments are available under the Annotate menu. You can add, edit or delete a comment for the following annotations: image stamp, arrow, line, rectangle, ellipse, text, freehand, stamp and highlight.

The following example shows comments displayed in the Viewer:



Note that you can hide or show the comments by clicking on the Comments Panel icon located in the upper right-hand corner of the Viewer:

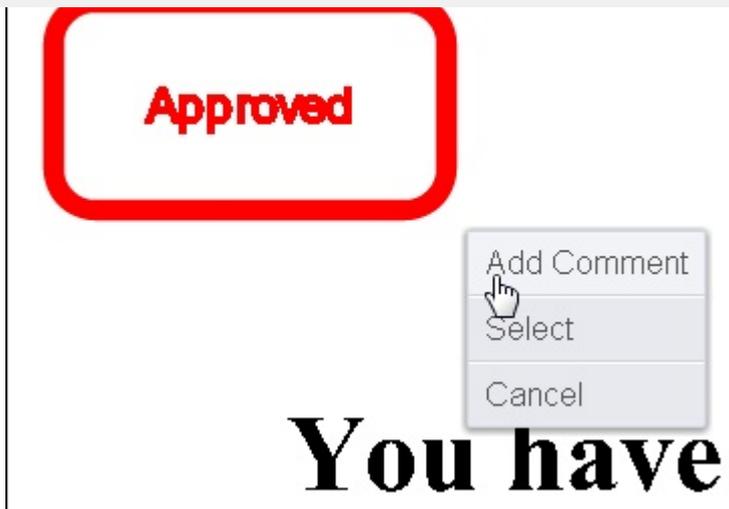


### To Add a Comment to an Annotation

1. Select the **annotation** you want to use and draw it on the document.
2. A **mini-content menu** immediately displays near the bottom right of the annotation:



3. Hover over the **mini-context menu** and the following options are displayed: **Add Comment**, **Select** or **Cancel**:



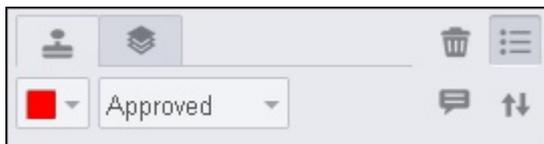
4. Select **Add Comment**. A text box displays to the right of the document:

A screenshot of a comment input box. It features a text area with the placeholder text "Type comment here". Below the text area are two buttons: "Cancel" and "Done".

5. Type in your **comment** and click **Done**.

### To Add a Comment to an Existing Annotation

1. Click on an **existing annotation**. A context menu displays to the right of the document:



2. Select the **Comment icon**. A text box displays to the right of the document:

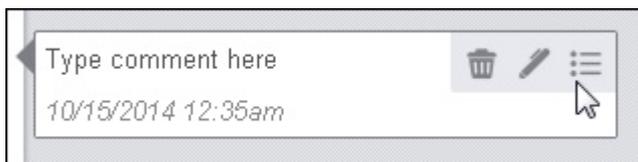
A screenshot of a comment input box. It features a text area with the placeholder text "Type comment here". Below the text area are two buttons: "Cancel" and "Done".

3. Type in your **comment** and click **Done**.

### To Edit a Comment

1. Click on an **existing annotation**. The comment box displays to the right of the document:

A screenshot of a comment input box. It features a text area with the placeholder text "Type comment here". Below the text area are two buttons: "Cancel" and "Done".



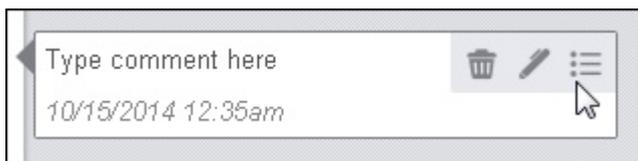
3. Make your edits to the **comment** and click **Done**.

### To Delete a Comment

1. Click on an **existing annotation**. The comment box displays to the right of the document:



2. Hover over the **context menu** on the comment box:



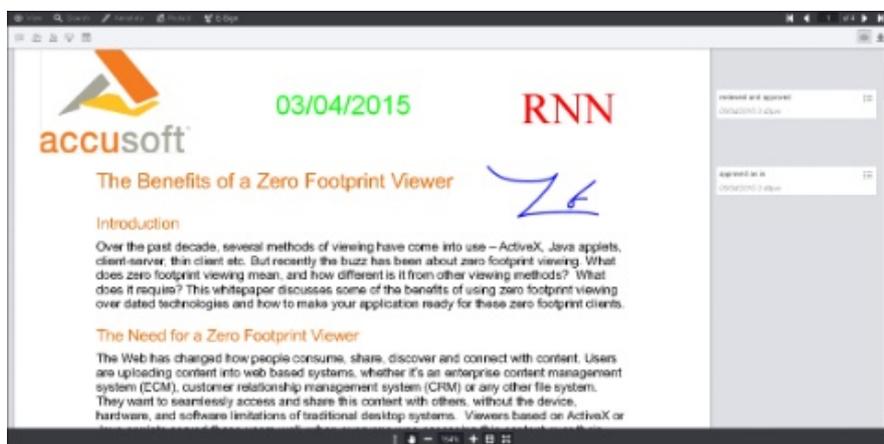
3. Click on the **Trash icon** to delete the comment.

## Use E-Signature Comments

Comments are available under the E-Signature menu. You can add, edit or delete a comment for the following eSignatures: Freehand, Text and Place Date.

 When in Horizontal Layout Mode, comments are not supported in the Viewer.

The following example shows comments displayed in the Viewer:



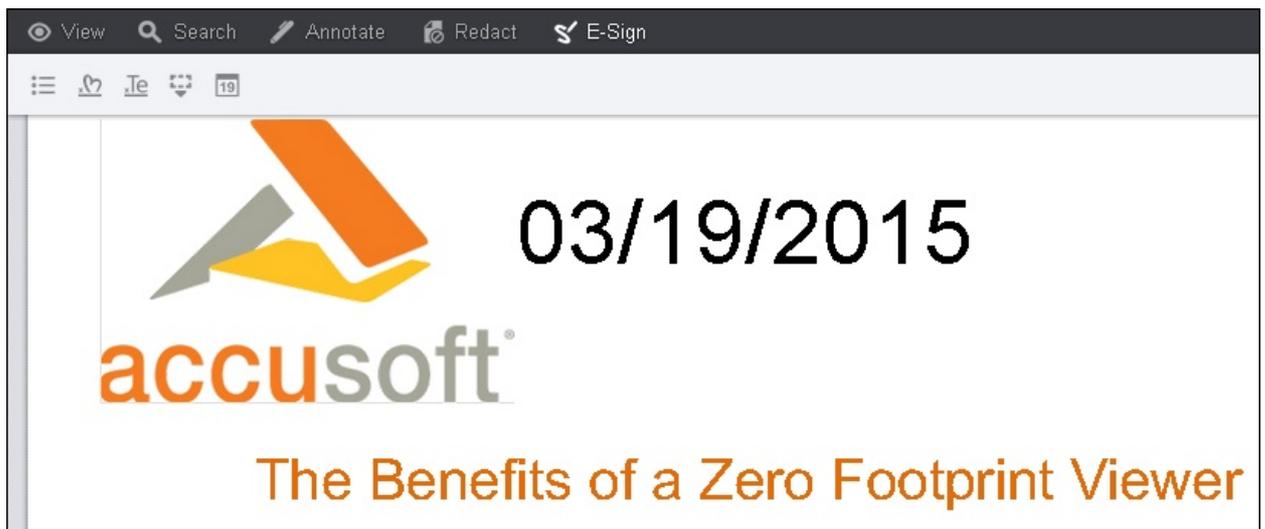
Note that you can hide or show the comments by clicking on the Comments Panel icon located in the upper right-



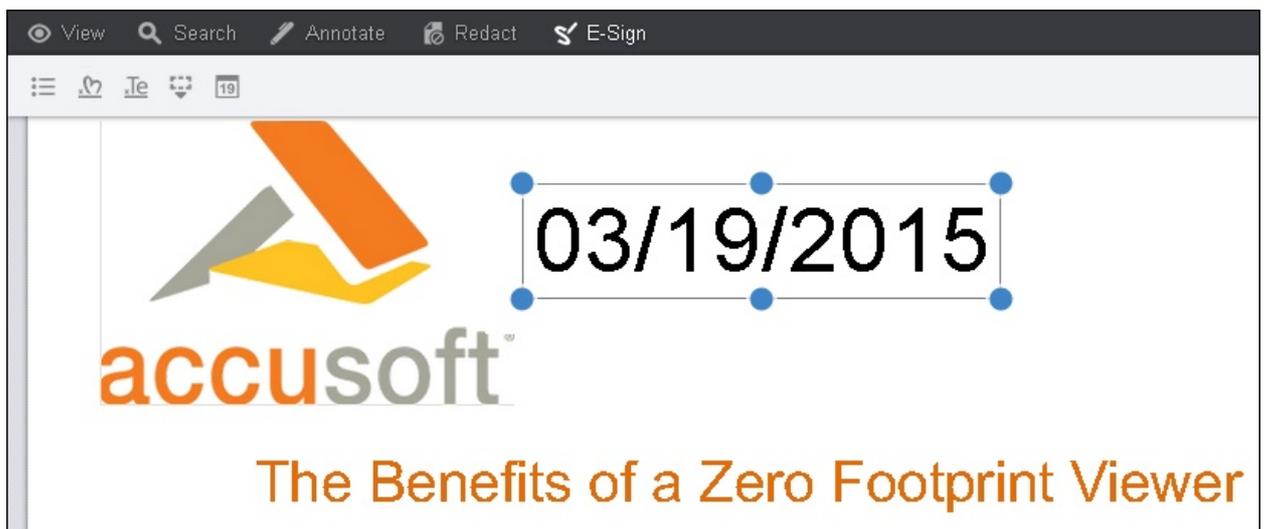
## To Add a Comment to an E-Signature

This section uses the **Place Date** E-Signature as an example. You can also add comments to the Freehand and Text eSignatures.

1. Select the **Place Date** E-Signature you want to use and draw it on the document:



2. Click on the **Date** and the comment box appears to the right:



3. Click on the **Comment** icon in the comment box. A text box displays:

### To Add a Comment to an Existing E-Signature

1. Click on an **existing E-Signature**. A context menu displays to the right of the document:



2. Select the **Comment icon**. A text box displays to the right of the document:



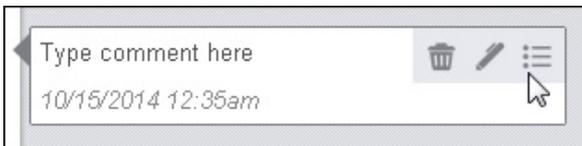
3. Type in your **comment** and click **Done**.

### To Edit a Comment

1. Click on an **existing E-Signature**. The comment box displays to the right of the document:



2. Hover over the **context menu** on the comment box. Click on the **Pen icon**:



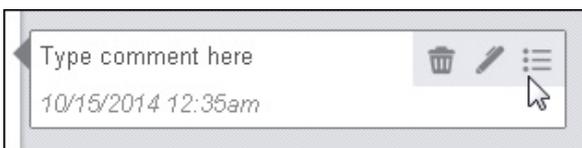
3. Make your edits to the **comment** and click **Done**.

### To Delete a Comment

1. Click on an **existing E-Signature**. The comment box displays to the right of the document:



2. Hover over the **context menu** on the comment box:



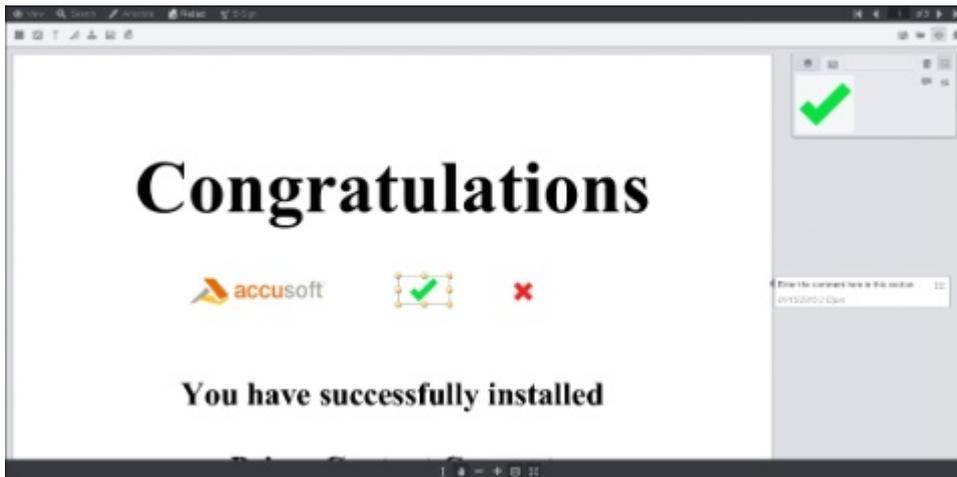
3. Click on the **Trash icon** to delete the comment.

### USE REDACTION COMMENTS

Comments are available under the Redact menu. You can add, edit or delete a comment for the following redactions: image stamp, filled rectangle, transparent rectangle, text, stamp, and text selection.

 When in Horizontal Layout Mode, comments are not supported in the Viewer.

The following example shows comments displayed in the Viewer:



Note that you can hide or show the comments by clicking on the Comments Panel icon located in the upper right-hand corner of the Viewer:

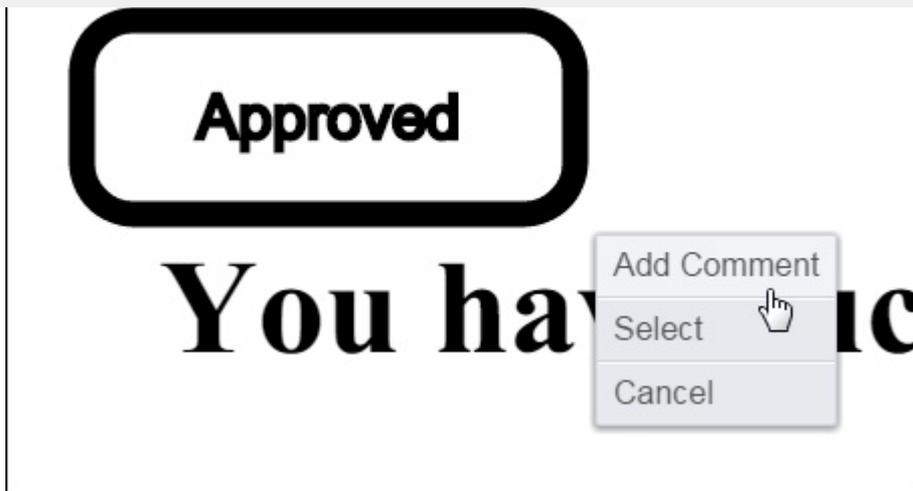


#### To Add a Comment to a Redaction

1. Select the **redaction** you want to use and draw it on the document.
2. A **mini-content menu** immediately displays near the bottom right of the redaction:



3. Hover over the **mini-context menu** and the following options are displayed: **Add Comment**, **Select** or **Cancel**:



4. Select **Add Comment**. A text box displays to the right of the document:

5. Type in your **comment** and click **Done**.

### To Add a Comment to an Existing Redaction

1. Click on an **existing redaction**. A context menu displays to the right of the document:

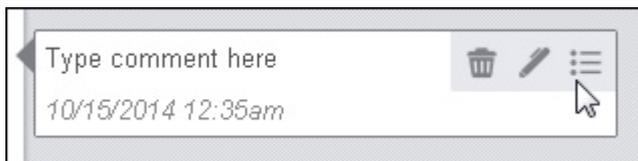


2. Select the **Comment icon**. A text box displays to the right of the document:

3. Type in your **comment** and click **Done**.

### To Edit a Comment

1. Click on an **existing redaction**. The comment box displays to the right of the document:



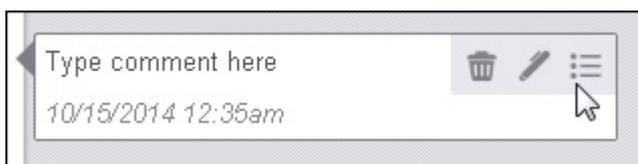
3. Make your edits to the **comment** and click **Done**.

### To Delete a Comment

1. Click on an **existing redaction**. The comment box displays to the right of the document:



2. Hover over the **context menu** on the comment box:

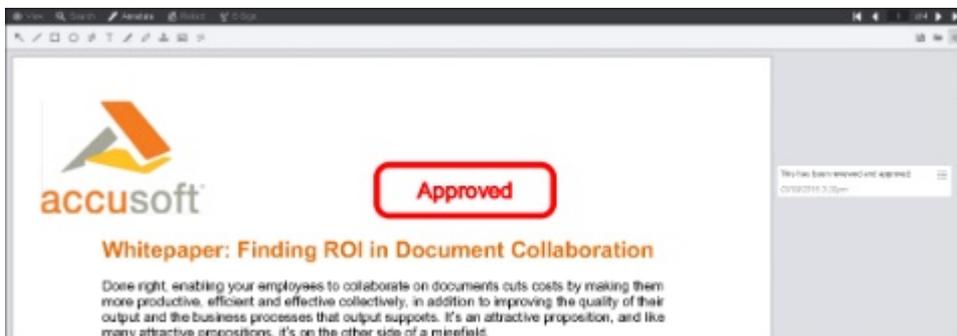


3. Click on the **Trash icon** to delete the comment.

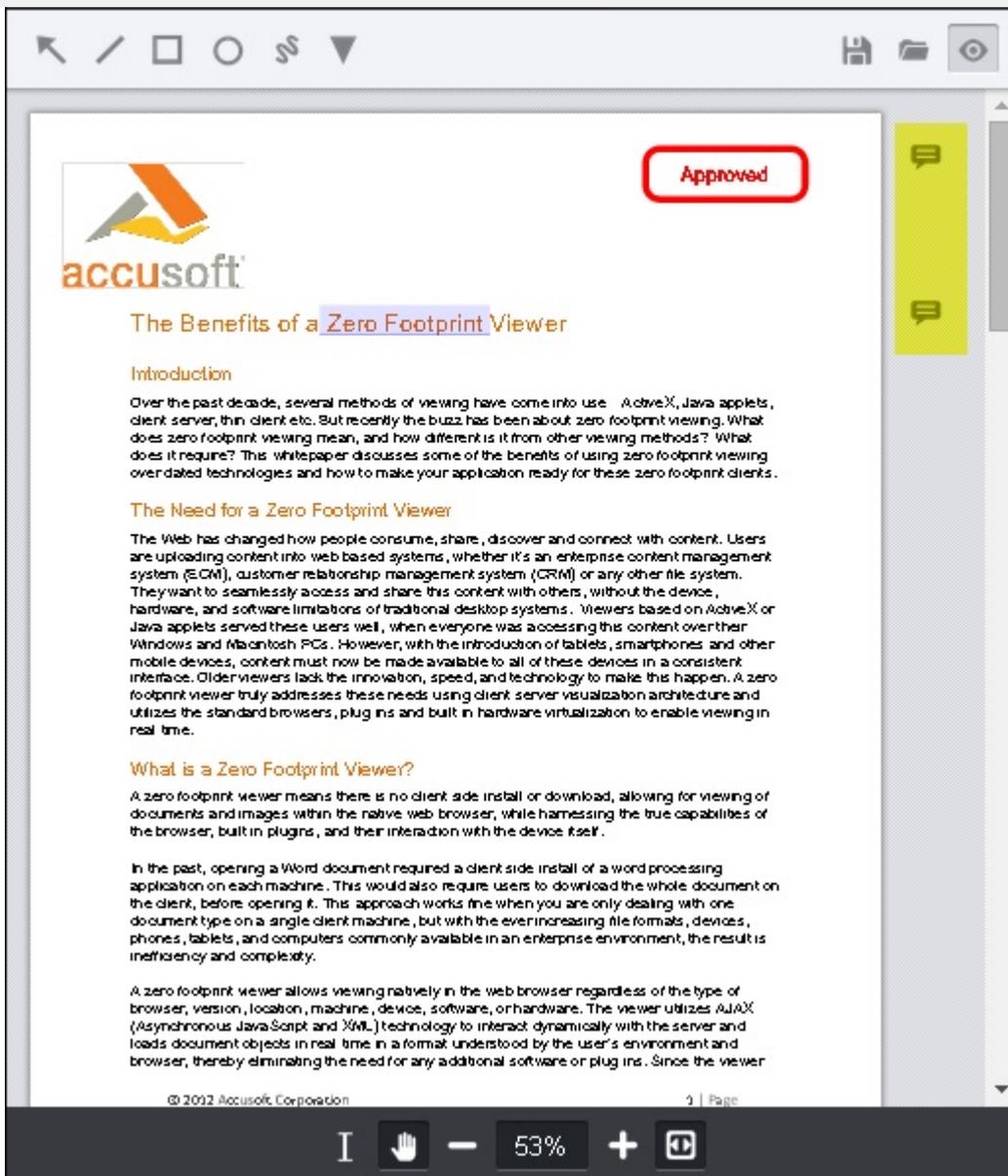
## Use Skinny Comments

By default, the Viewer automatically chooses to display either the Skinny or Full Comments panel based on the size of the Viewer pane. Additional components, such as whether or not the Search or Thumbnail panels are open, can affect which Comments panel is displayed. For more information on customizing the behavior of the Comments panel, refer to the topic: [Configuring the Skinny Comments panel](#).

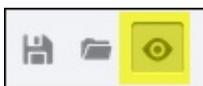
An example of the Full Comments panel:



An example of the Skinny Comments panel highlighted in yellow below:

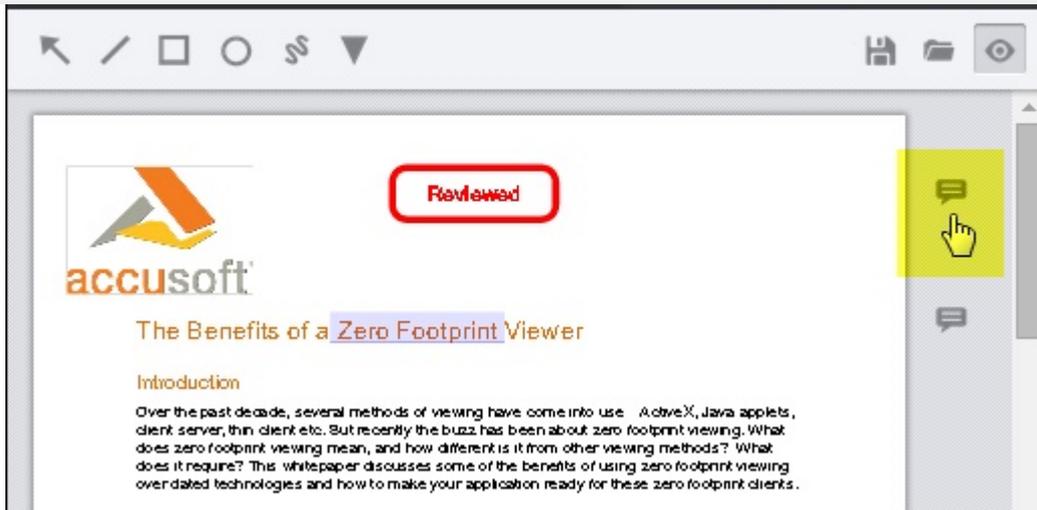


Note that you can hide or show the comments by clicking on the Comments Panel icon located in the upper right-hand corner of the Viewer:

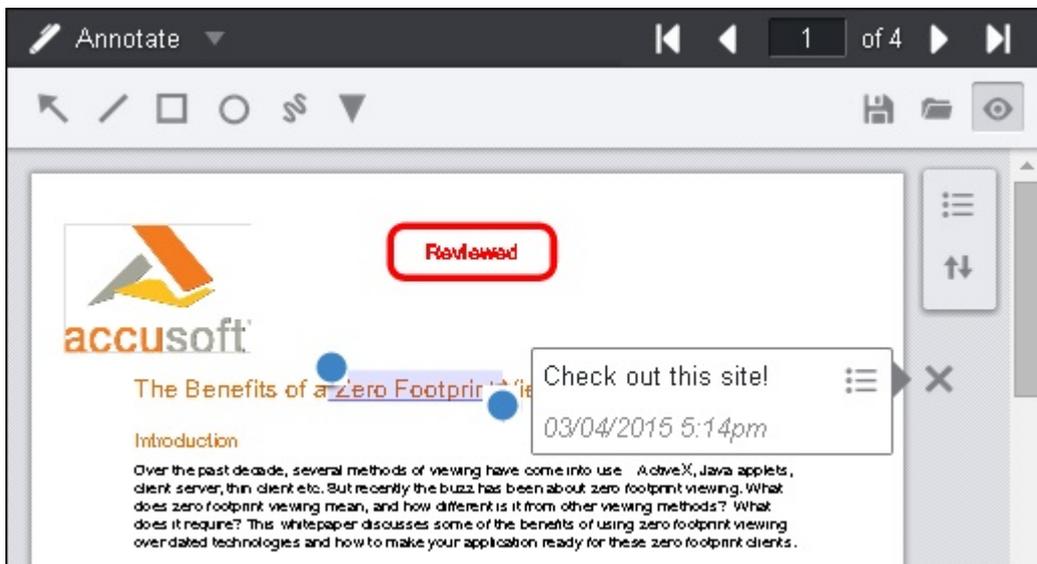


### To view or edit Skinny comments

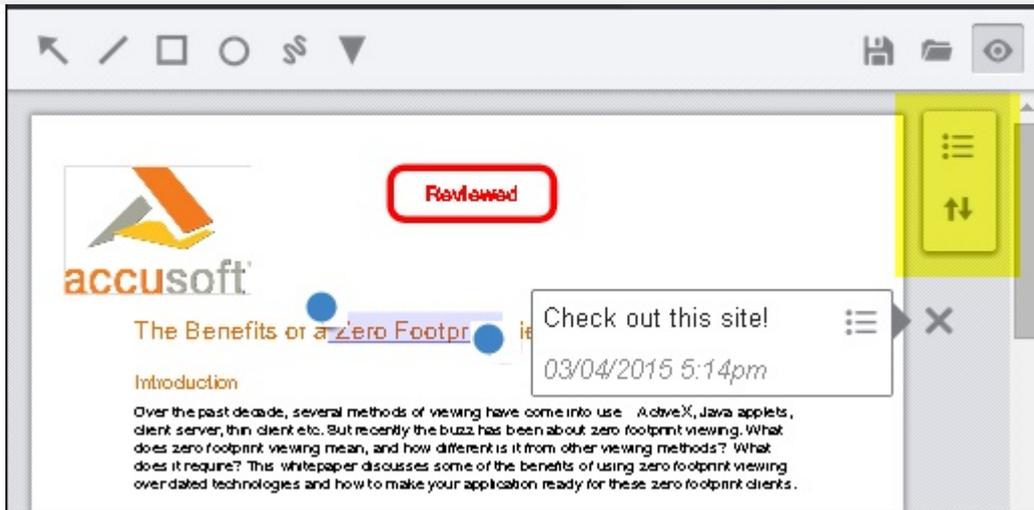
1. Click on the **Comment icon**:



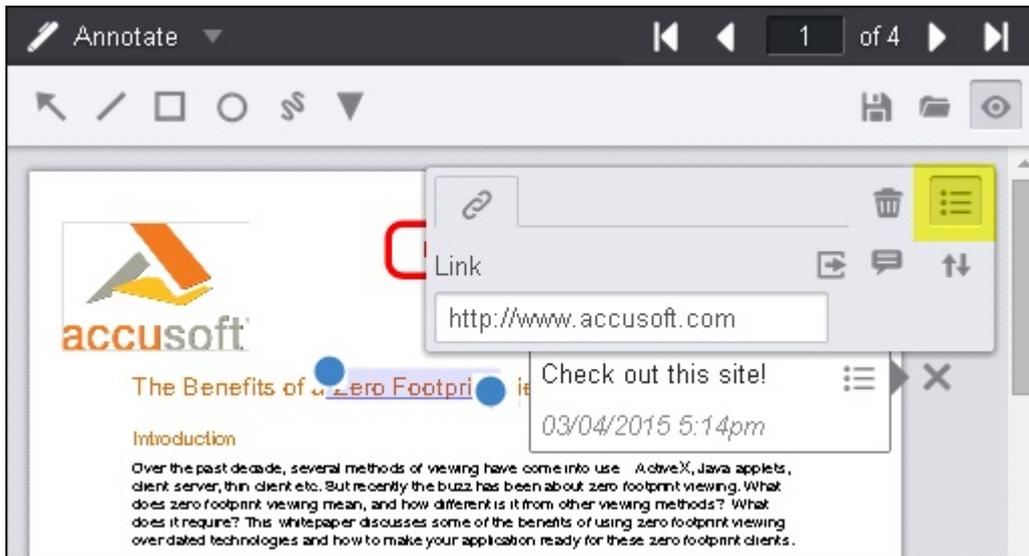
The comment box is displayed:



2. The **Options** and **Move Menu** icons display next to the comments:

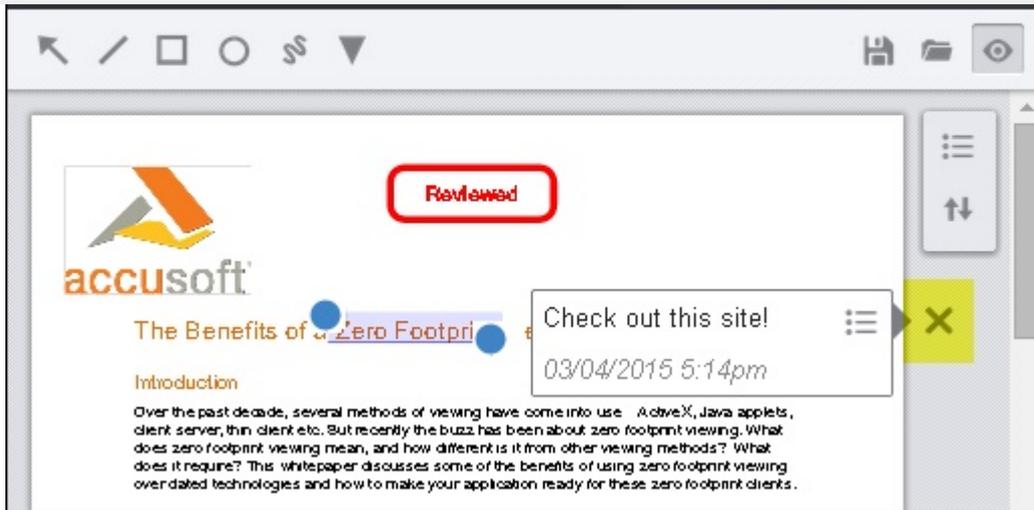


3. Click on the **Options icon** and the Options context menu is displayed:



Make the desired changes.

4. Click on the **"X" icon** and the Option menu is minimized:



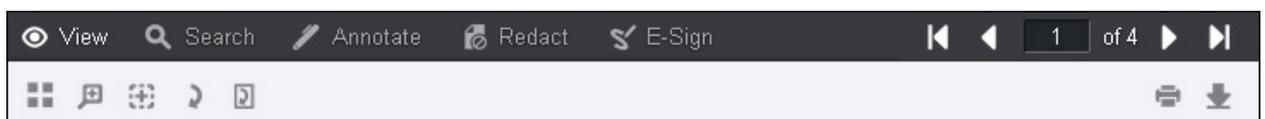
## Print Documents

The Viewer enables you to print documents using various settings:

- Page Range
- Page Orientation
- Paper Size
- Auto Fit Margins
- Annotations
- Redactions
- Headers, Footers and Margins
- Comments
- Redactions Reasons

### To print a document:

1. To open the print menu, click on the **Print** button:



2. Select your print **options**:

Pages

Print All

Print Current

1-5, 8, 10-15

Show...

Annotations

Redactions

Review Redacted Text

Less options ▲

Orientation

Portrait ▼

Paper Size

Letter ▼

Auto Fit Margins

Comments

Do Not Print ▼

Redaction Reasons

Do Not Print ▼

Print

3. Click **Print**.

## Additional Printing Information

### Page Range

Within the print menu, a user may specify a single page to print or multiple pages. Individual pages and page ranges need to be separated by a comma (,). Pages within a range should be separated by a dash (-). Below are sample values that may be entered into the print-range input field:

- 1
- 1, 3, 5, 8
- 1-3, 5, 8
- 1-3, 5, 8-10

### Page Orientation

This option enables the user to set the document orientation. When a document is loaded into the Viewer, the default page orientation (Portrait or Landscape) is based on the dimension of the first page loaded. For example, if the dimension of the first page is Landscape, the Viewer will set the Landscape property in the viewer's Print menu.

 Users are required to manually set the page orientation within the system's Print Dialog menu. Even when a particular orientation has been set within the Viewer, the user will have to set the orientation within the system's Print Dialog menu once the content has been sent to the printer. Currently, Google Chrome is the only browser that is capable of detecting the appropriate page orientation.

### Paper Size

You can select the desired paper size from the Paper Size drop-down.

 Users are required to manually set the page size within the system's Print Dialog menu. Even when a particular page size has been set within the Viewer, the user will have to set the page size within the system's Print Dialog menu once the content has been sent to the printer. Currently, Google Chrome is the only browser that is capable of detecting the appropriate page size.

This option chooses whether or not the Viewer should use the default browser margins. This is usually an issue with the Internet Explorer and Safari browsers:

- **On** - When necessary, the pages will be smaller, so that the entire page content can fit on one printed page. No extra user action is needed to adjust the margins. This is the default setting.
- **Off** - Content will always be printed as an 8.5 x 11 inch page. The user is expected to set the browser print margins to 0. See the section for "Headers, Footers, and Margins" below.

## Annotations

This option chooses whether or not to include annotations in the printed document.

 Printing with annotations is not supported in Internet Explorer 8.

 Firefox does not support printing white or gray text by default; the text is instead printed black. To print white or gray text you must select the **Print Background (colors & images)** option in Firefox. You can do so by opening the Firefox menu, selecting **Print**, clicking the **Page Setup...** button, checking the **Print Background (colors & images)** option, and clicking the **OK** button.

## Redactions

This option chooses whether or not to include redactions in the printed document. You can also choose to review the redacted text by selecting the Review Redacted Text checkbox.

 Firefox does not support printing white or gray text by default; the text is instead printed black. To print white or gray text you must select the **Print Background (colors & images)** option in Firefox. You can do so by opening the Firefox menu, selecting **Print**, clicking the **Page Setup...** button, checking the **Print Background (colors & images)** option, and clicking the **OK** button.

## Headers, Footers and Margins

The Viewer's printing control is designed to optimize for the amount of space available on each piece of paper. The Viewer defines the margins within the CSS, however some settings within the user's browser may need to be altered in order to properly format the pages for printing. The following user actions need to use the "Auto Fit Margins: off" option:

### Internet Explorer

In order to maximize the space available on each sheet of paper, verify that the Header and Footer have been set to "Empty" and that "Print Background Colors and Images" has been checked within the browsers "Page Setup". To verify your settings in IE, go to Settings > Print > Page Setup.

### Safari

In order to maximize the space available on each sheet of paper, verify that the "Print Headers and Footers" checkbox is not selected in the browser's print menu. Under the "Paper Size" dropdown menu, select "Manage Custom Sizes." Configure the top, bottom, left, and right margins to be 0. Use this paper size profile to print without margins.

### Chrome

No user actions are needed. Chrome provides the latest CSS3 specification for printing, which allows disabling print margins and header and footer content. This option is always used by the Viewer when printing in Chrome.

### Opera

No user actions are needed. Opera provides the latest CSS3 specification for printing, which allows disabling print margins and header and footer content. This option is always used by the Viewer when printing in Opera.

No user actions are needed. Firefox provides a native API to disable print margins and header and footer content. This option is always used by the Viewer when printing in Firefox.

### Comments

You can select to have the Comments in the document print after each page, all at the end of the document or not print at all.

### Redactions Reasons

You can select to have the Redaction Reasons in the document print after each page, all at the end of the document or not print at all.

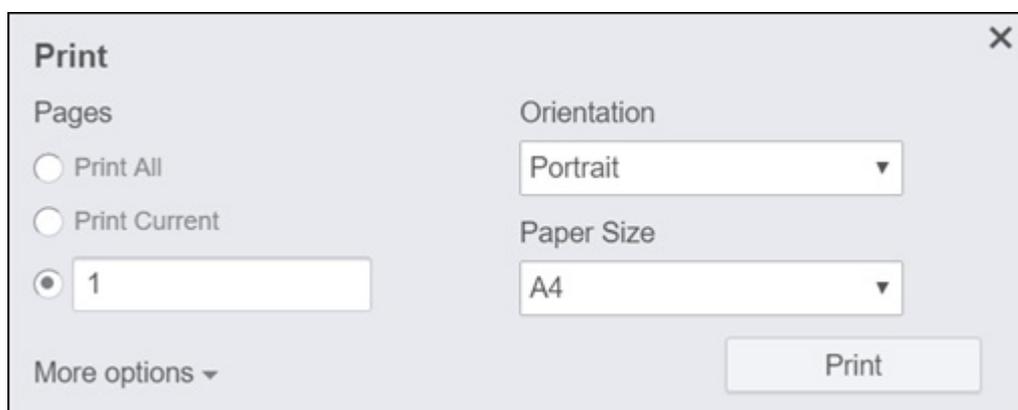
## Print Non-standard Size Documents

Printing documents of non-standard paper sizes inside the Viewer can sometimes give you unwanted results such as cropping of pages or some empty spaces at the bottom of the printing area. Let's look at why such problems may occur and the printing workflow you should use inside the Viewer in order to eliminate those issues and achieve the best printing results. Printing a document in the Viewer is a two-step process which differs from a standard printing scenario that you might see in some desktop applications. This section discusses why this two-step process is required.

No matter what format your document is in currently, in order to display it in the browser it has to be converted into SVG format. The caveat of using SVG is that it's scalable and doesn't have a fixed width or height properties embedded into it (for example, like the PNG image). That's why in order to print it correctly we need to apply a specific size before sending it to the printing tool that is part of your browser. Let's look at how this can be done.

### Step 1: Forcing a specific print size

1. Open a **document** of choice inside the Viewer.
2. Click **Print** and the Print dialog displays:

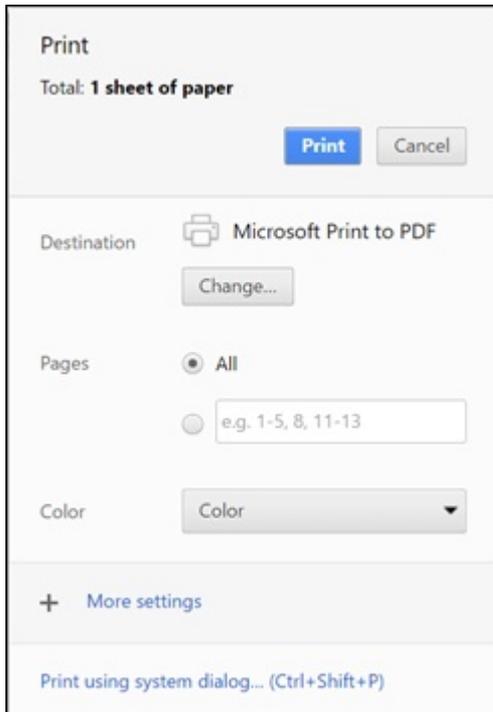


3. Select the **printing options** that you need. The first page of the document will print in portrait orientation using A4 paper size. (That's where we force the SVG content to be a specific printing size).

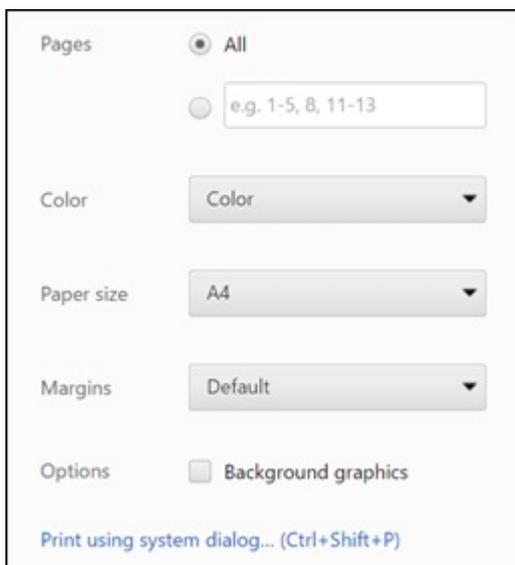
## Step 2: Sync browser printing settings

The Print Preview dialog is a part of the browser and will differ depending on what browser you use. The following example uses Google Chrome.

1. Once all of the printing data has been loaded, a Print Preview dialog displays the default options:



2. Since you already selected the print settings in Step 1 above, now you need to make sure that the browser print settings are the same. Click on **More Settings**.
3. The list of options is similar to the Viewer Print dialog. Select the **A4 paper size** from the drop-down:



4. The non-standard paper sizes should now print correctly.

size, refer to the [Printing custom paper sizes code example](#).

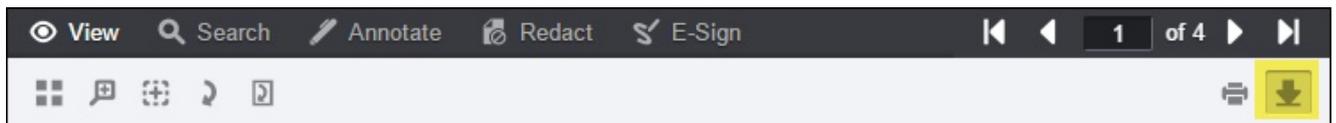
## Download Documents

This section explains how to download a document, including how to include some or all annotations, redactions, and e-signatures, as well as how to use the preview mode:

- [Downloading the Original Document](#)
- [Burning Annotated Content in the Viewer](#)
- [Burning Redacted Content in the Viewer](#)
- [Burning Signed Content in the Viewer](#)

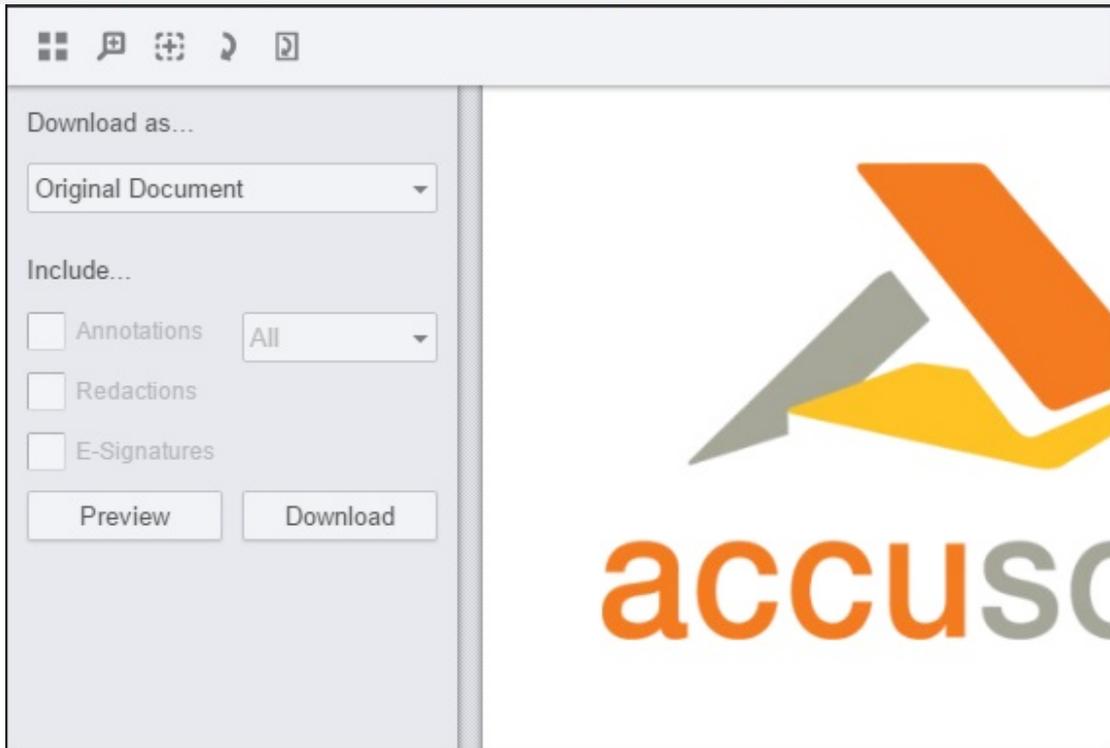
## Downloading the Original Document

The Download button is available under the View, Annotate, Redact and E-Sign menus:

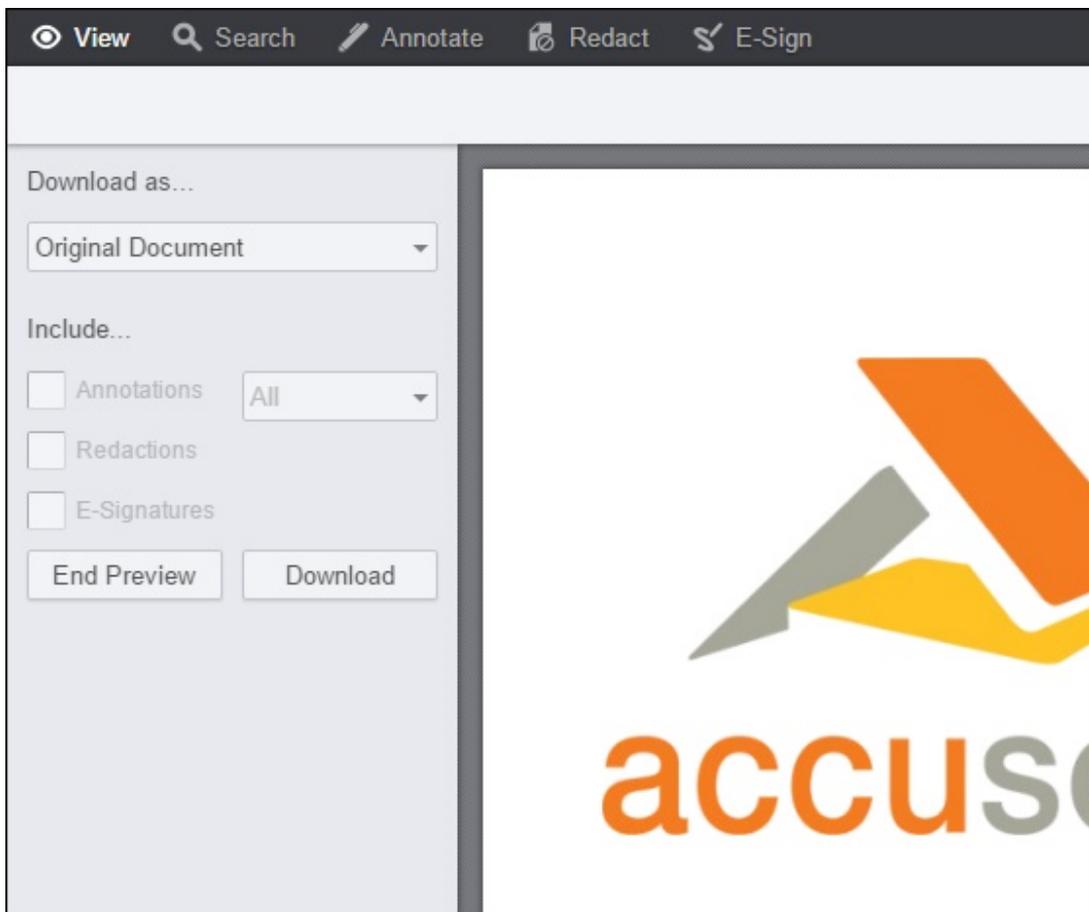


**To download the original document:**

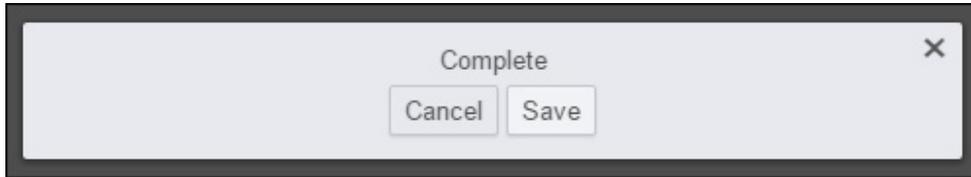
1. Click on the **Download** button. The Download as... properties panel displays:



2. If you want to preview the document before downloading, click on **Preview**. The document is displayed in Preview Mode:



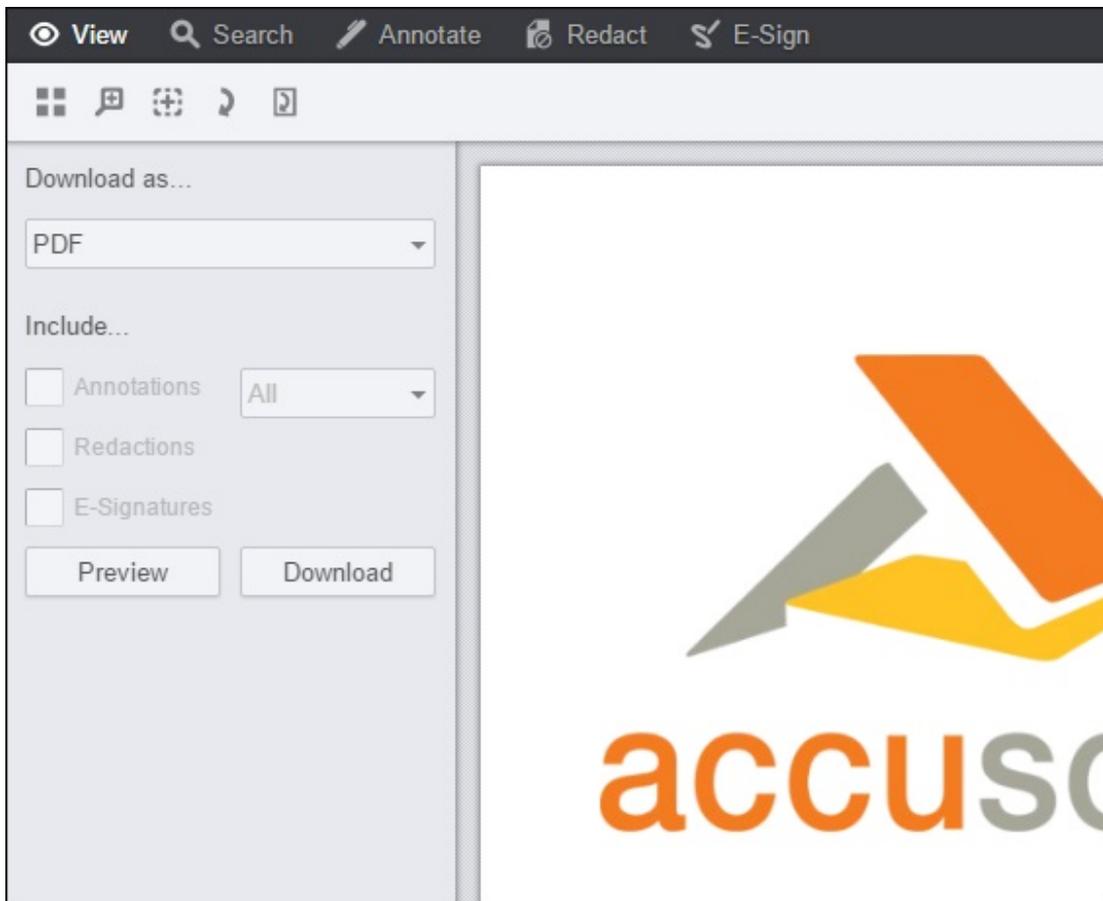
3. Click on **Download** and the Complete dialog box displays:



4. Click **Save** and a copy of the original document is saved to your PC.

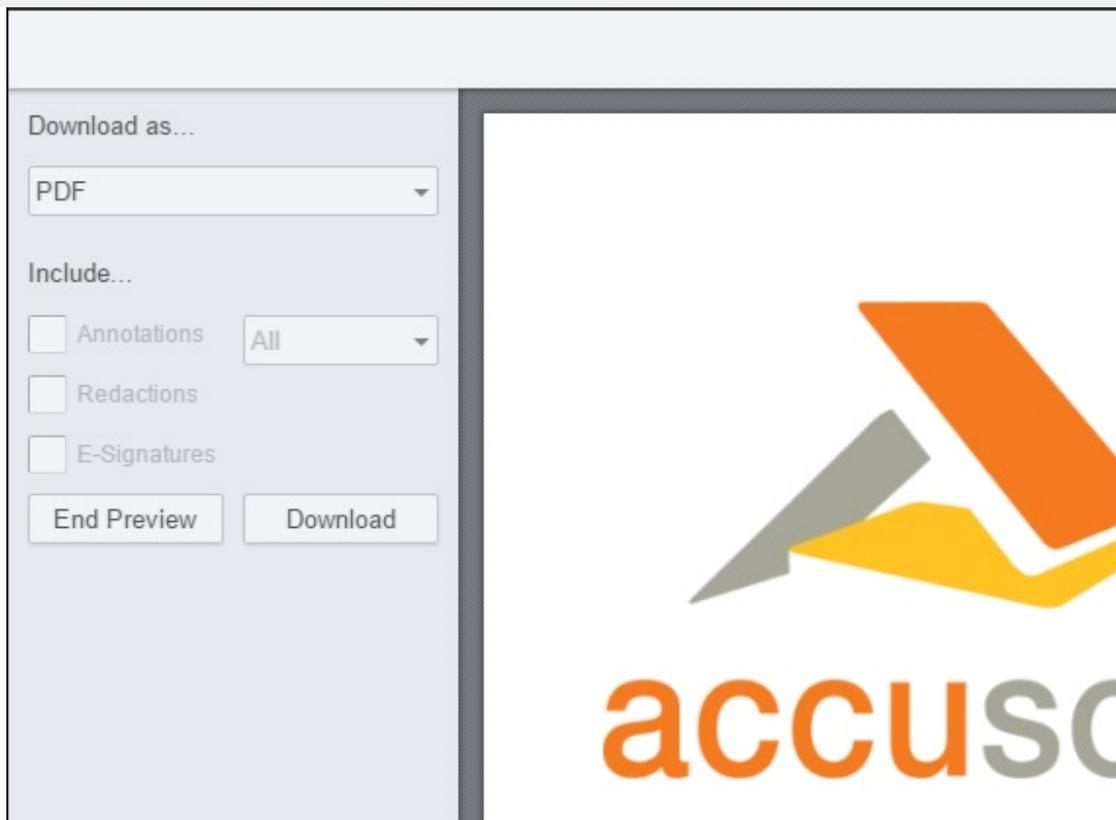
### To download a PDF of the original document:

1. Click on the **Download** button. The Download as... properties panel displays:



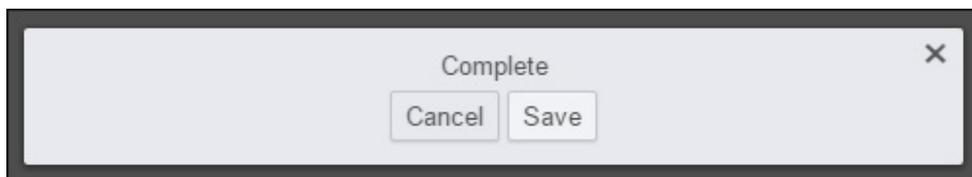
Select **PDF** from the drop-down menu.

2. If you want to preview the document before downloading, click on **Preview**. The document is displayed in Preview Mode:



To close Preview Mode, click on **End Preview**.

3. Click on **Download** and the Complete dialog box displays:

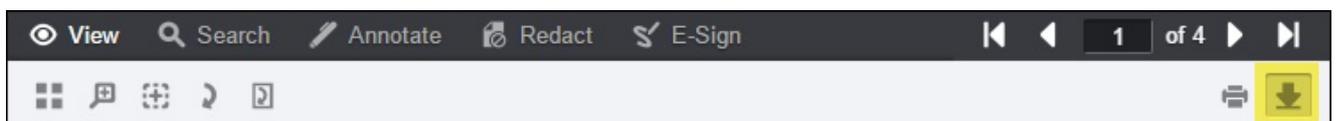


4. Click **Save** and a PDF of the original document is saved to your PC.

## Burning Annotated Content in the Viewer

PrizmDoc allows you to download your annotated document directly from the Viewer. The document download icon will present you with additional options when you have annotations to burn-in to the document.

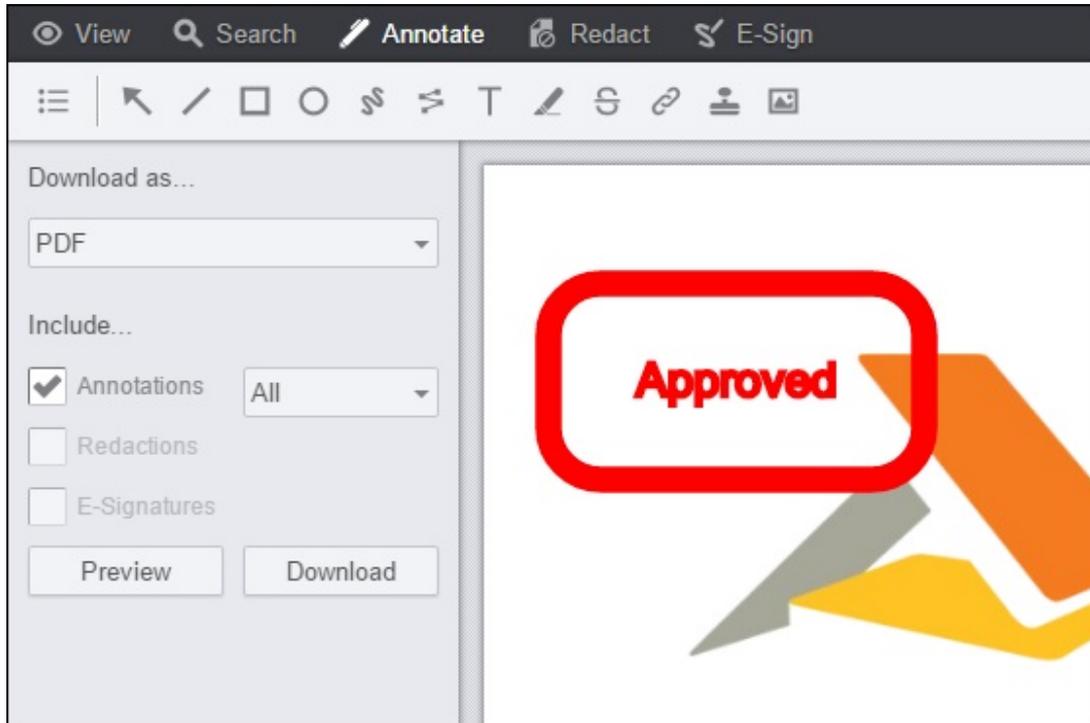
The Download button is located on the menu bar on the right-hand side of the Viewer:



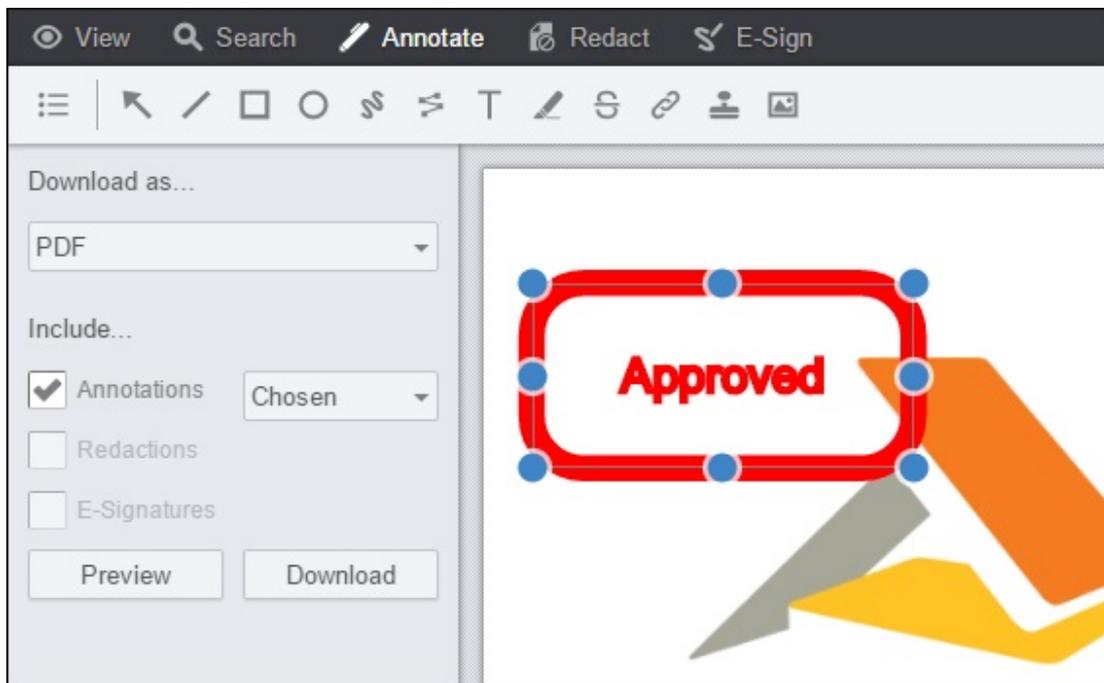
### To burn in annotated content:

The document download icon will present you with additional options when you have annotated content to burn-

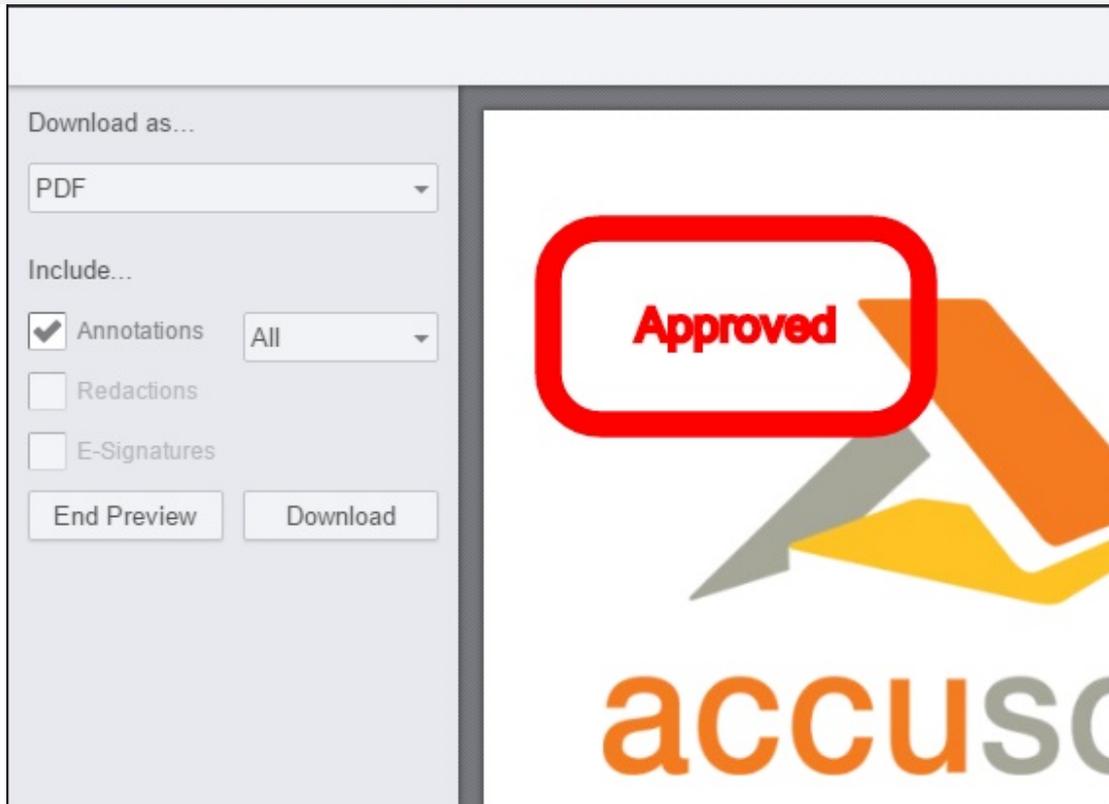
1. To download your annotated document, click on the **Download** button and the properties panel displays:



2. Select the **Annotations** checkbox. You can select **All** or **Chosen** from the drop-down menu, depending on which annotations you want to burn-in:

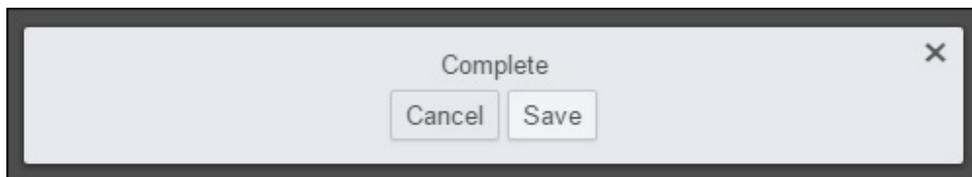


3. If you want to preview the document before downloading, click on **Preview**. The document is displayed in Preview Mode:



To close Preview Mode, click on **End Preview**.

4. Click on **Download** and the Complete dialog box displays:



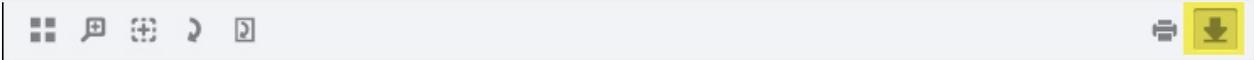
5. Click **Save** and the markup burner process will burn the annotation on the server and download a new, annotated PDF document.

Note that the annotated document will be downloaded to the browser following your browser's document download process. The original document will be maintained in the Viewer and will not be affected by the burn-in process. For more information on mark up burners, refer to [Working with the PrizmDoc Server > API Reference > PrizmDoc RESTful API > Markup Burners](#).

## Burning Redacted Content in the Viewer

PrizmDoc allows you to download your redacted document directly from the Viewer. The document download icon will present you with additional options when you have redactions to burn-in to the document.

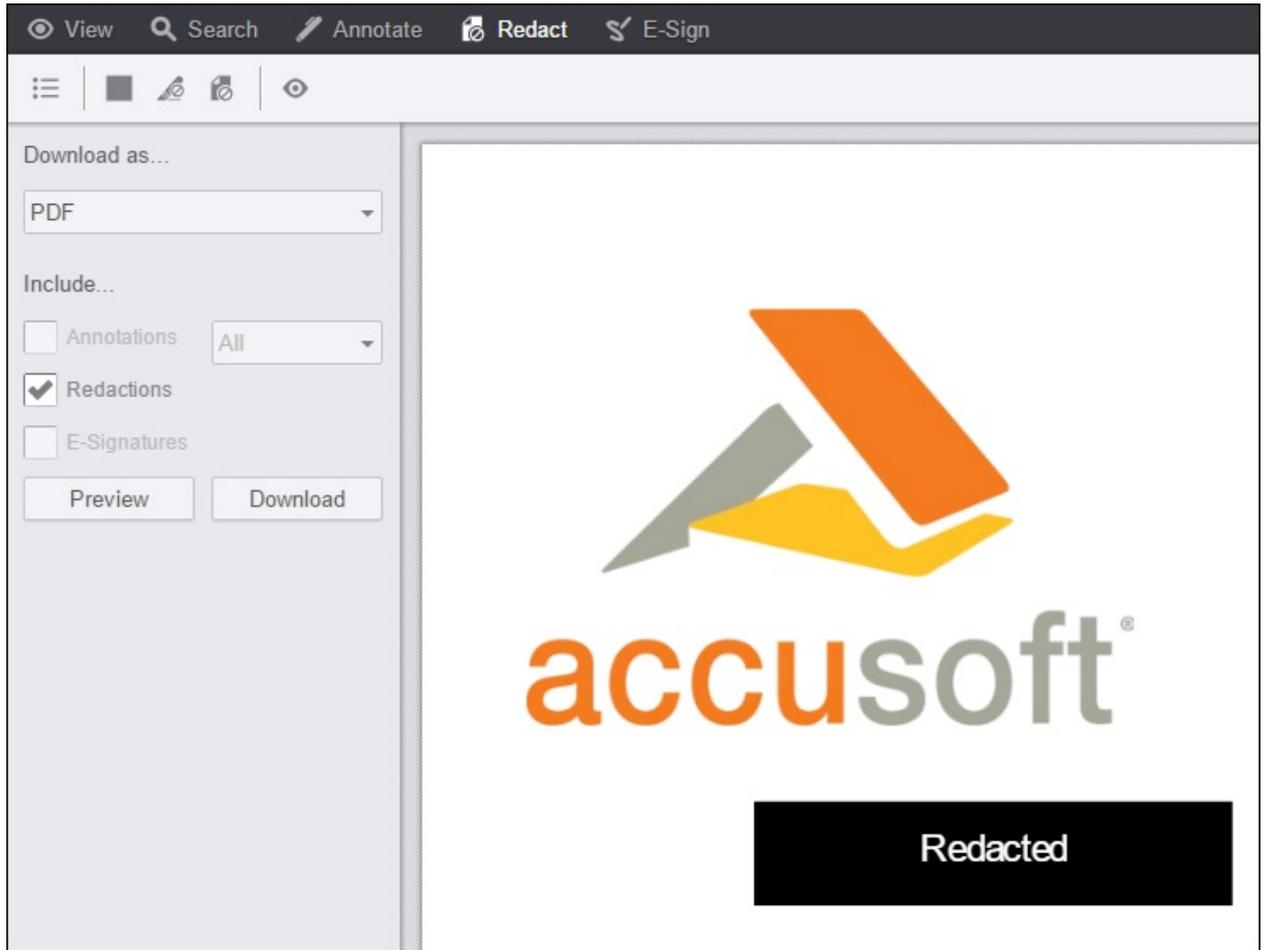
The Download button is located on the menu bar on the right-hand side of the Viewer:



### To burn in redacted content:

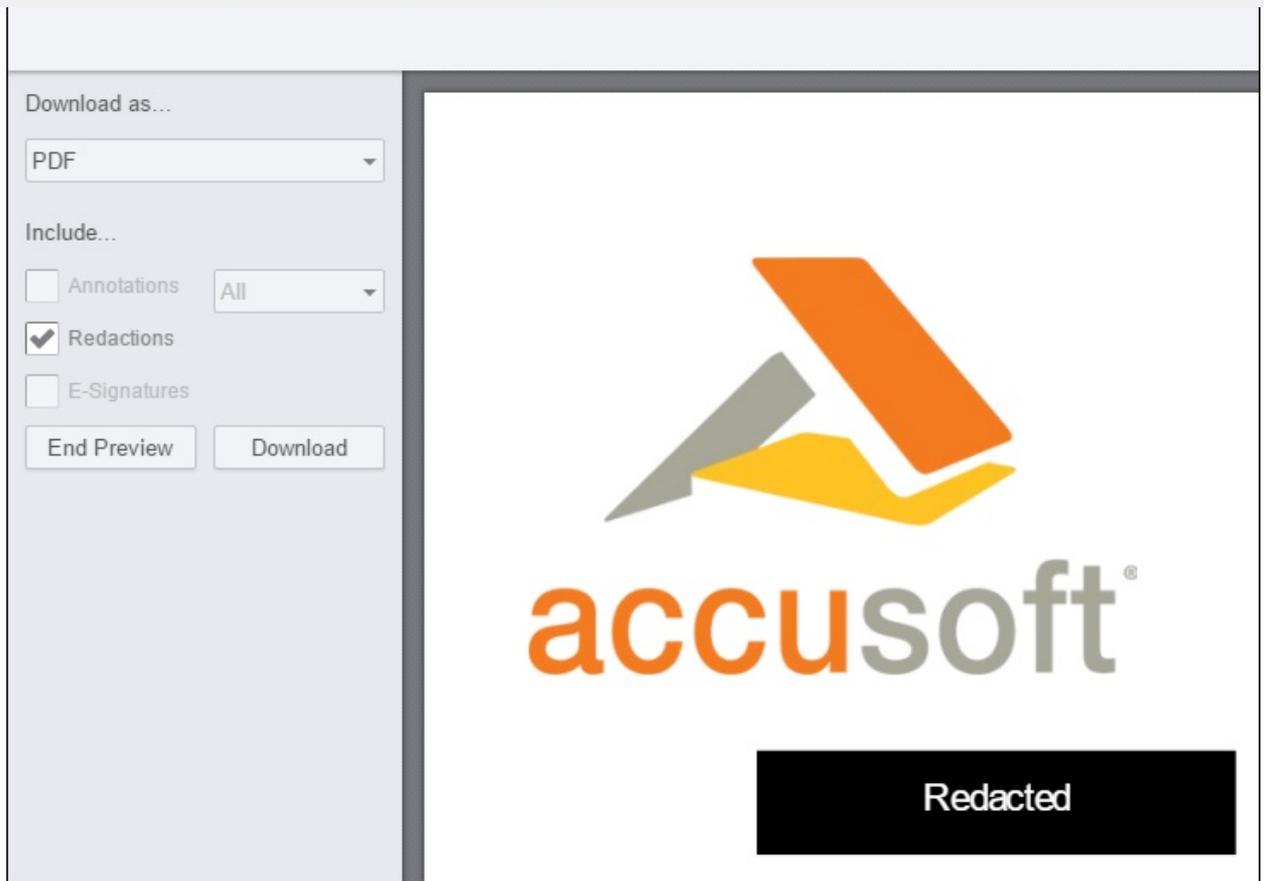
The document download icon will present you with additional options when you have redacted content to burn-in to the document. Redactions do not need to be saved prior to initiating the redaction burn-in process.

1. To download your redacted document, click on the **Download** button and the properties panel displays:



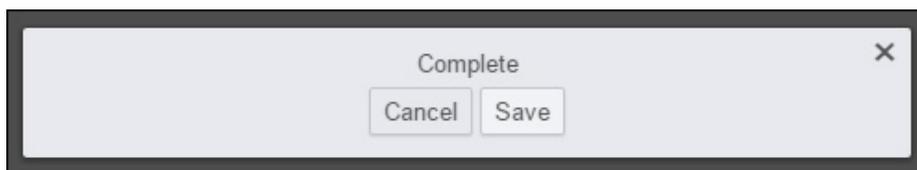
Select the **Redactions** checkbox.

2. If you want to preview the document before downloading, click on **Preview**. The document is displayed in Preview Mode:



To close Preview Mode, click on **End Preview**.

3. Click on **Download** and the Complete dialog box displays:



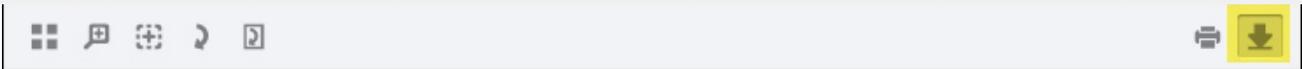
4. Click **Save** and the markup burner process will burn the redaction on the server and download a new, redacted PDF document.

Note that the redacted document will be downloaded to the browser following your browser's document download process. The original document will be maintained in the Viewer and will not be affected by the burn-in process. For more information on mark up burners, refer to [Working with the PrizmDoc Server > API Reference > PrizmDoc RESTful API > Markup Burners](#).

## Burning Signed Content in the Viewer

PrizmDoc allows you to download your signed document directly from the Viewer. The document download icon will present you with additional options when you have e-signatures to burn-in to the document.

The Download button is located on the menu bar on the right-hand side of the Viewer:



### To burn in signed content:

The document download icon will present you with additional options when you have an e-signature to burn-in to the document. E-signatures do not need to be saved prior to initiating the e-signature burn-in process.

1. To download your signed document, click on the **Download** button and the properties panel displays:



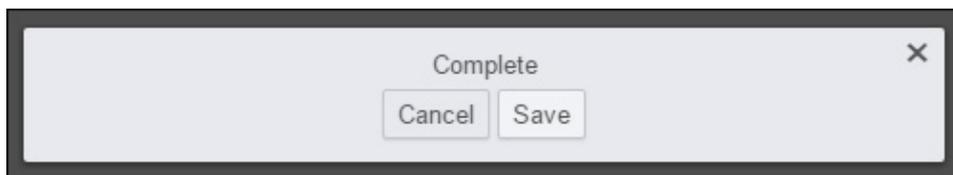
Select the **E-Signatures** checkbox.

2. If you want to preview the document before downloading, click on **Preview**. The document is displayed in Preview Mode:



To close Preview Mode, click on **End Preview**.

3. Click on **Download** and the Complete dialog box displays:



4. Click **Save** and the markup burner process will burn the e-signature on the server and download a new, signed PDF document.

Note that the signed document will be downloaded to the browser following your browser's document download process. The original document will be maintained in the Viewer and will not be affected by the burn-in process. For more information on mark up burners, refer to [Working with the PrizmDoc Server > API Reference > PrizmDoc RESTful API > Markup Burners](#).

## Reference

This section contains additional information on using the Viewer features:

- [Tools](#)
  - [Viewer Main Screen Options](#)

- [Search Tab](#)
- [Annotate Tab](#)
- [Redact Tab](#)
- [E-Sign Tab](#)
- [Keyboard Shortcuts](#)
- [Work with Sticky Mouse Tools](#)

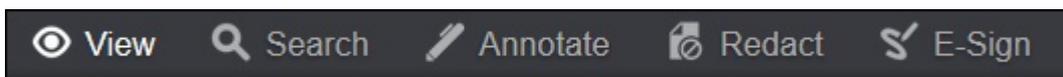
## Tools

A listing of all the tools in the Viewer, what they can do, and their corresponding icon:

- [Viewer Main Screen Options](#)
- [View Tab](#)
- [Search Tab](#)
- [Annotate Tab](#)
- [Redact Tab](#)
- [E-Sign Tab](#)
- [Keyboard Shortcuts](#)
- [Work with Sticky Mouse Tools](#)

## Viewer Main Screen Options

From the main screen in the Viewer, the following tabs are available for viewing and working with your documents:



Click on the links below to read more about the features you can use on each of the tabs:

- [View Tab](#)
- [Search Tab](#)
- [Annotate Tab](#)
- [Redact Tab](#)
- [E-Sign Tab](#)

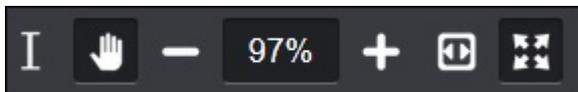
You can navigate through your document using the following features that are always available from each tab:

### Document Navigation from the Top of the Screen



For more details on these options, refer to [View Tab > Document Navigation from the Top of the Screen](#).

## Document Navigation from the Bottom of the Screen

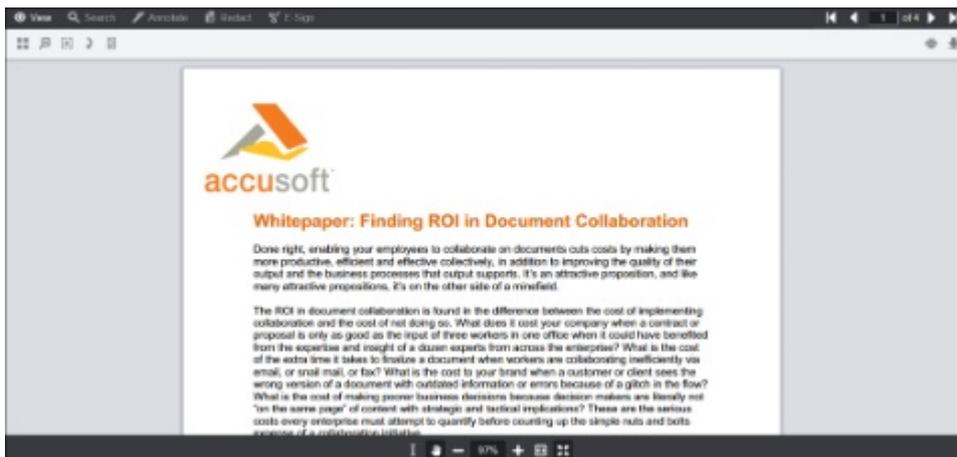


- Select Text, Pan Tool, Zoom Out, Zoom Percentage, Zoom In, Fit Content, and Full Screen

For more details on these options, refer to [View Tab > Document Navigation from the Bottom of the Screen](#).

## View Tab

The View Tab is the default tab that is displayed when you first open a document in the Viewer. Once you've loaded a document, it should look similar to this:



You will immediately note that you can scroll the page up and down via either the mouse or touch, just like any web page you've ever worked with. As you scroll through the document, each page becomes visible until you reach the last page in the document.

In addition to scrolling, there are several other Viewer adjustments that can be made for reading a document. These are located at the top and bottom of the browser window, and they allow you to easily adjust the page to your viewing needs. The following section provides you with more information on these features.

## Document Navigation from the Top of the Screen

The following table shows the icons and their meanings:

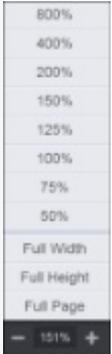
	<p><b>Thumbnails</b> - The Thumbnail panel provides an overview of the document and your current position in it. Scrolling through the document is easy via this panel, and by simply clicking on a specific page you can load that page into the Viewer. Also, by adjusting the slider at the top of the thumbnail pane you can make the thumbnail images smaller or larger.</p>
---	---

	specific portions of the document, similar to using a magnifying glass. By clicking the mouse and then dragging it across the document, you can see an enlargement of the area directly under the mouse cursor.
	<b>Rectangle Zoom</b> - The Rectangle Zoom tool makes it easy to zoom in on an arbitrary rectangle in the document. Simply click and drag out the portion of the document you wish to zoom in on and the document zoom percentage will be adjusted accordingly. Unlike the Magnifier, the zoom level is set by the rectangle you draw, and once set in this way, will remain in effect until you modify it.
	<b>Rotate Document</b> - The Rotate Document tool makes it easy to rotate all the pages in the document by 90 degrees. To continue to rotate all the pages in the document by another 90 degrees, continue to press the Rotate Document button.
	<b>Rotate Page</b> - The Rotate Page tool allows you to rotate one page 90 degrees without rotating all the pages in the document. Simply scroll to the page you want to rotate and select the Rotate Page tool. To continue to rotate that page another 90 degrees, continue to press the Rotate Page button.
	<b>First Page</b> - Select the First Page button to go to the very first page of the document.
	<b>Previous Page</b> - Select the Previous Page button to go back one page from your current location in the document.
	<b>Next Page</b> - Select the Next Page button to go forward one page from your current location in the document.
	<b>Last Page</b> - Select the Last Page button to go to the very last page of the document.
	<b>Print</b> - When you click on the Print button, a dialog is displayed that allows you to choose the Pages, Orientation, Paper Size, Auto Fit Margins, Annotations, Redactions, Comments and Redaction Reasons you would like to print. For more details on printing, refer to <a href="#">Printing Documents</a> .
	<b>Download</b> - When you click on the Download button, a panel is displayed that allows you to choose the Original Document or PDF and if you want any Annotations, Redactions or E-Signatures to print as well. For more details on downloading documents and the options available, refer to <a href="#">Downloading Documents</a> .

## Document Navigation from the Bottom of the Screen

The navigation buttons at the bottom of the screen remain available through all pages of the document that you are viewing. The following table shows the icons and their meanings:

	<b>Select Text</b> - The Select Text tool allows you to select text and perform actions such as copying it to the clipboard or highlighting it.
---	---

	<p>called the Pan tool, and when it's active (when the button appears indented) you are in "Pan mode" in the Viewer, which is the "base" state where you can select and work with items. If you're ever attempting to select an item and it draws an annotation instead, it's because this tool isn't currently selected. Many times this mode will be selected automatically for you. For example, the default behavior after drawing an annotation using one of the annotation tools is to return immediately to Pan mode so the newly created annotation can be manipulated as desired.</p>
	<p><b>Zoom Out</b> - The Zoom Out tool allows you to reduce the document size in the viewing area so you can easily see the borders of the document.</p>
	<p><b>Viewing Options</b> (Percentage, Full Width, Full Height, Full Page) - The Viewing Options field displays your current zoom level (in this example, the area that displays 151%). Click the Viewing Options field to open a menu for additional viewing options.</p>
	<p><b>Zoom In</b> - The Zoom In tool allows you to enlarge the document in the viewing area so it is easier to read.</p>
	<p><b>Fit Content</b> - The Fit Content tool allows you to quickly expand the document to fit the width of the viewing area, with minimal borders on each side of the document.</p>
	<p><b>Full Screen</b> - The Full Screen tool allows you to expand the document to fit the width of the entire Viewer, with no visible borders on each side of the document.</p>

## Working with Large Files

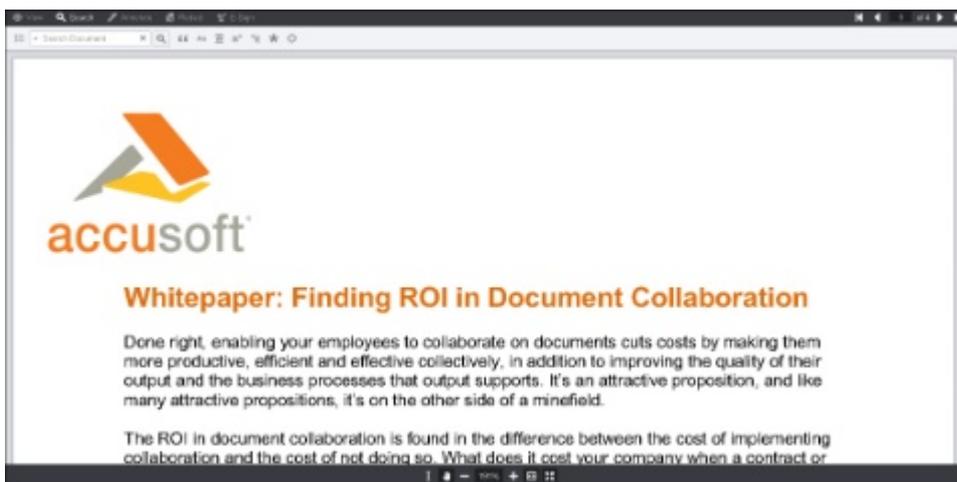
Note that if you open, scroll-through or use set page number for a very large file, you may see the spinning circle (highlighted in yellow below) displayed in the middle of the Viewer while the document loads:



## Search Tab

The Viewer contains powerful search features which allow you to look for various types of text within text-enabled documents such as Word documents and PDF files. In addition to searching within the text of your document, you can now perform a powerful set of document review tasks, using expanded Search features including a proximity search. A proximity search can be performed using the ~n syntax, where "n" is how many words can be between the first and second search terms and still return a result. With support for annotations search, you can search in comments, text annotations, redaction reasons, as well as see all of the annotations created in your document. Annotations can be navigated to by viewing annotation search results in the review panel or by navigating to each annotation in the document.

Example of the Search tab in the Viewer:



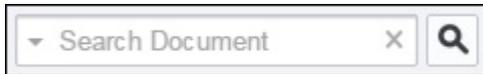
This section contains the following information:

- [Text Search Features](#)
- [Search Icons](#)
- [Search Filters & Document Review Features](#)

### Text Search Features

## Search Document Text Box

Search by entering text in the **Search Document** text box:



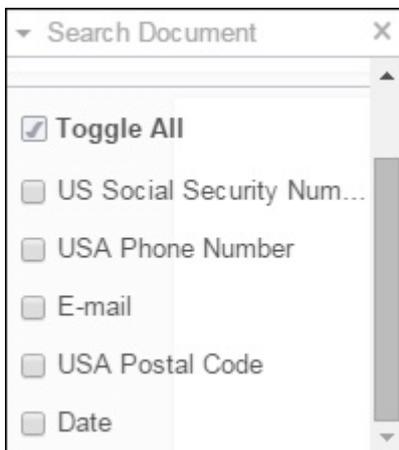
## Search Icons

Narrow your search by selecting the **Search** icons:



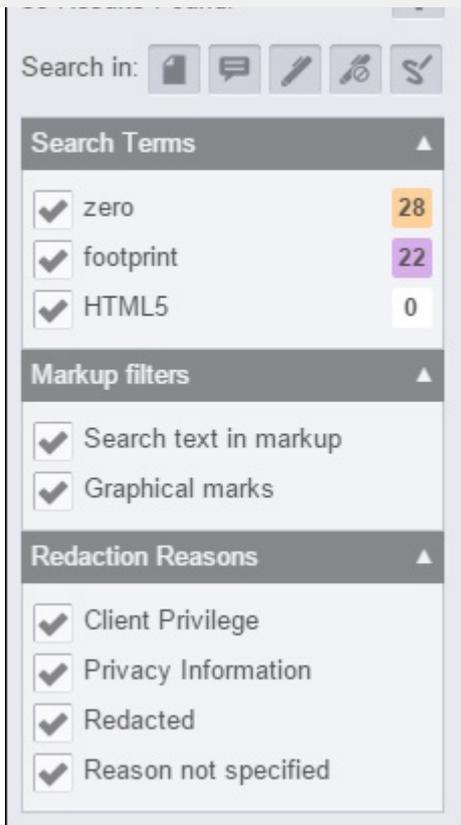
## Search Patterns

Search using the **Search Patterns** drop-down list:



## Search Filter

Search by using the advanced **Search Filter**:



 If you want to create a set of predefined search terms, refer to the topic, [How to use the Predefined Search](#).

## Search Icons

The Search icon bar:



You can search using the following criteria:

	<b>Match Exact Word or Phrase</b> - For example, you can search for an exact word "home" or a phrase "go home".
	<b>Match Case</b> - For example, you can search for "Home" or "home".
	<b>Whole Word</b> - For example, you can search for "home" and the search results will return "home" and "homeschooled".
	<b>Begins With</b> - For example, you can search for "con" and the search results will return "content" and "continued".
	<b>Ends With</b> - For example, you can search for "ent" and the search results will return "document" and "content".

search results will return all email addresses that have "@domain.com" in them (i.e., jane@domain.com). You can also use the question mark (?) and search for "ra?e" and the search results will return "rate", "rake", "race", etc.

 When you select the 'Use Wildcards' icon, it automatically disables the following icons: 'Whole Word', 'Begins With' and 'Ends With'.

## Search Filters & Document Review Features

### Search Terms

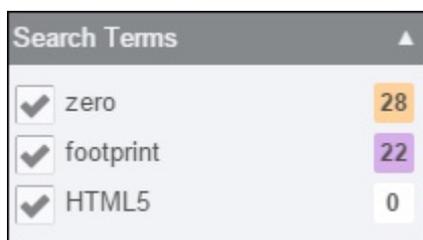
The search filter allows you to easily search any of the following in the document:

	<b>Searches for text</b> within the document that matches the specified search criteria.
	Searches for <b>text within comments</b> that contain the specified search criteria.
	Searches within <b>annotations</b> in the document. If selected, the annotation results returned will depend on the Markup filters selected, as shown below: <div data-bbox="269 1032 691 1216" style="border: 1px solid black; padding: 5px; margin: 10px 0;"><p>Markup filters ▲</p><p><input type="checkbox"/> Search text in markup</p><p><input type="checkbox"/> Graphical marks</p></div> <p><b>Search text in markup:</b> When the annotation filter is enabled, any text contained within the text annotation (and/or selected text within a highlight or strikethrough annotation), that matches the search criteria you enter, will be highlighted. The annotation will appear in the search results panel.</p> <p><b>Graphical marks:</b> When the annotation filter is enabled and Graphical marks is selected, all annotations including graphical marks (lines, arrows, freehand, etc.) will be returned in the search results. This is a convenient way to quickly review all the annotations in your document.</p> <p> The annotation button must be selected for the Markup filters to return any content.</p>
	Searches all <b>redactions</b> in the document. The redactions filter's results are independent of the search terms entered, because redacted content is not searched. However, by selecting the redactions filter, you can see and navigate to all redactions in the document. You can also filter Redaction Reasons by looking for redactions with specific reasons, or no reason at all. Redaction Reasons are configurable. See <a href="#">Using Redaction Reasons</a> for more information.

<input checked="" type="checkbox"/> Client Privilege <input checked="" type="checkbox"/> Privacy Information <input checked="" type="checkbox"/> Redacted <input checked="" type="checkbox"/> Reason not specified	
<input checked="" type="checkbox"/> 	Finds all <b>signatures</b> in the document. This is independent of search criteria. The search does not inspect the content of the signature, but simply allows you to quickly find and navigate to the 'non-burned-in' signatures in the document.

## Search Terms Count

The Search Terms pane displays an aggregated list of search results, sorted by most to least hits found. In the following example, "zero" appears 28 times in the document:

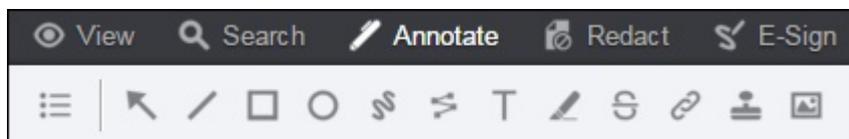


If multiple search terms are selected, search hits can be further toggled in the results list.

## Annotate Tab

### Annotate Tab

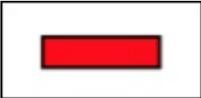
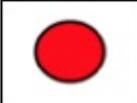
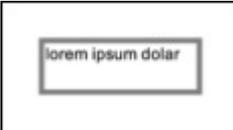
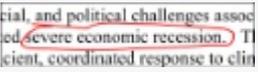
The Annotate tab contains components that allow you to annotate a document. Note that annotations can be burned into the document, but they do not remove any of the underlying text content. The following image shows the Annotate tab within the Viewer:

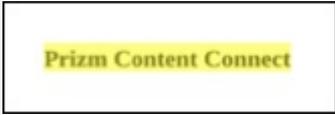
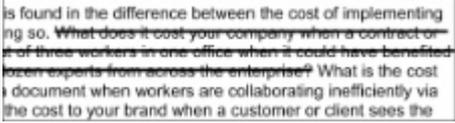
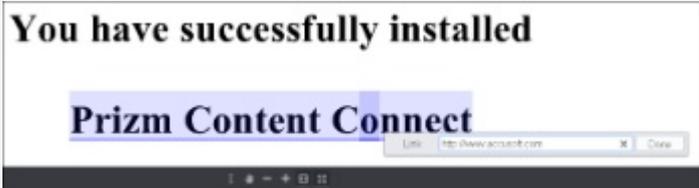


 You can easily modify your hyperlink and highlight annotations by using either touch or mouse.

The Annotate tab contains tools that allow users to annotate a document. Below is a description of the annotation toolbar:

Button Name	Description & Example
Annotation Layers	Use to view annotation layers:

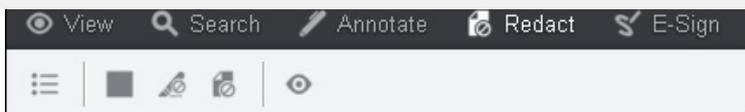
<p>Image Stamp Annotation</p>	<p>Use to select image stamp annotations:</p> 
<p>Arrow Annotation</p>	<p>Use to draw arrow annotations:</p> 
<p>Line Annotation</p>	<p>Use to draw line annotations:</p> 
<p>Rectangle Annotations</p>	<p>Use to draw rectangle or square annotations:</p> 
<p>Ellipse Annotation</p>	<p>Use to draw ellipse or circular annotations:</p> 
<p>Text Annotation</p>	<p>Use to write text annotations:</p> 
<p>Freehand Annotation</p>	<p>Use to draw freehand annotations:</p> 
<p>Stamp Annotation</p>	<p>Use to draw stamp annotations:</p> 

Highlight Annotation	
Strikethrough Annotation	Use to strikethrough text on the document: 
Text Hyperlink Annotation	Use to create a hyperlink on the document: 
Polyline Annotation	Use to draw Polyline annotations: 
Edit Annotation Button	Use to select and alter the properties of annotations that have been generated or loaded in the Viewer.
Select Text Button	Use to select text in the document.
Pan Button	Use to pan around the document.
Save Annotations Button	Use to save the annotations for later use.
Load Annotations Button	Use to load previously saved annotations.

## Redact Tab

### Redact Tab

The Redact tab contains components that allow you to redact a term or whole pages in a document. Note that redactions can be burned into the document and they will remove the underlying text content. You also have the ability to search for a term and redact the search results - refer to the [Searching Documents > Redact Search Results](#) topic for more information.



You can easily modify your text selection redaction by using either touch or mouse.

You can hover over a redaction and see the content behind the redaction prior to burning it in. To review all the redacted content at once, click on the Redaction View Mode button on the toolbar.

The Redact tab contains tools that allow users to redact a document. Below is a description of the redact toolbar:

Button Name	Description & Example
Annotation Layers	<p>Use to view annotation layers:</p>
Filled Rectangle Redaction	<p>Use to draw solid, rectangle redactions:</p> <p>Comments Panel - choose fill color, line color, line width, text color:</p>
Text Selection Redaction	<p>Use to select and redact blocks of text using a text selection tool. After text selection, the highlight is converted to rectangle redactions. If multiple lines of text are selected, a rectangle redaction will be created for each line of selected text:</p> <p>Comments panel - choose the redaction reason:</p>

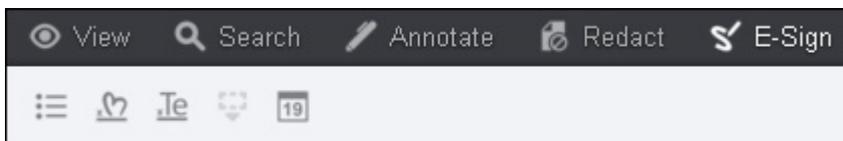
	<div data-bbox="304 277 793 342" style="border: 1px solid gray; padding: 2px;"> <span>Client Privilege</span> <span>↕</span> <span>↕</span> </div>
<p>Redact Full Pages</p>	<p>Use to redact the current page, all pages or a range of pages. Select and apply the redaction reason:</p> <div data-bbox="304 409 997 887" style="border: 1px solid gray; padding: 10px;"> <p><b>Redact Full Pages</b></p> <p><input type="checkbox"/> All pages</p> <p><input checked="" type="checkbox"/> Current page</p> <p><input type="checkbox"/> Page range <input type="text" value="1-5, 8, 10-15"/></p> <p>Reason <input type="text" value="Select..."/></p> <p><input type="button" value="Cancel"/> <input type="button" value="Redact"/></p> </div> <p>You can delete a full page redaction by clicking on the full page redaction and selecting Delete from the immediate action menu:</p> <div data-bbox="304 976 507 1149" style="border: 1px solid gray; padding: 5px;"> <p>Add Comment</p> <p>Delete </p> <p>Cancel</p> </div>
<p>Redaction View Mode</p>	<p>Use to see all redacted content:</p> <div data-bbox="304 1216 1489 1563" style="border: 1px solid gray; padding: 10px;"> <p><b>Whitepaper: Finding ROI in Document</b></p> <p>Done right, <span style="background-color: gray; color: black;">enabling your employees to collaborate on documents</span> cu more productive, efficient and effective collectively, in addition to imp</p> <p> You can print and review redacted text in a document by selecting the 'Review Redacted Text' checkbox on the Print dialog box. The Print button is available on the View menu.</p> </div>
<p>Select Text Button</p>	<p>Use to select text in the document.</p>
<p>Pan Button</p>	<p>Use to pan around the document.</p>
<p>Save Redactions Button</p>	<p>Use to save the redactions for later use.</p>
<p>Load Redactions Button</p>	<p>Use to load previously saved redactions.</p>

## E-Signatures

This topic contains the following information:

- [Creating Signatures](#)
- [Adding a Signature to a Document](#)
- [Creating Signature Categories](#)
- [How Signatures are Saved](#)

The E-Sign tab displays the Manage E-Signatures, Create Freehand Signature, Create Text Signature, Place E-Signature, and Place Date icons:



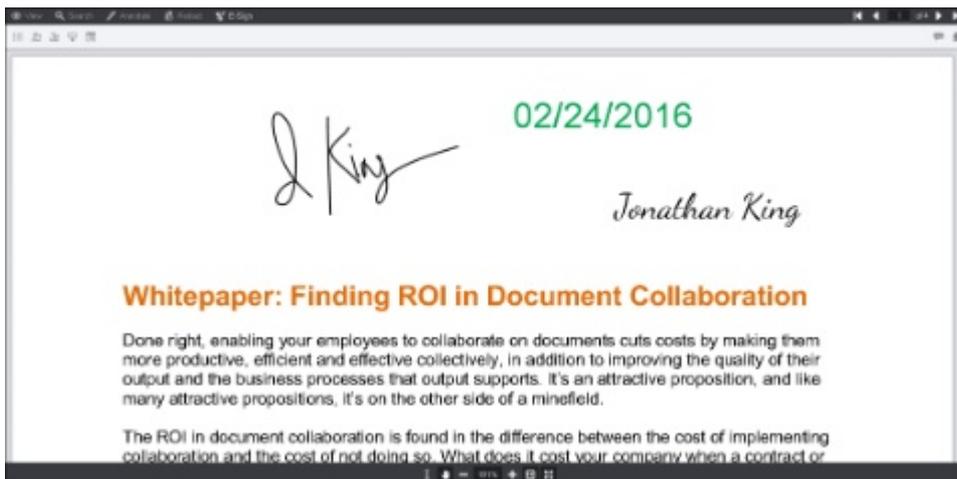
Hover over the icons for tooltips.

## Creating Signatures

You can create the following e-signatures:

- **Freehand** - draw your signature with your mouse
- **Text** - type your signature in a text box
- **Date** - select the date icon to apply the current date to your document

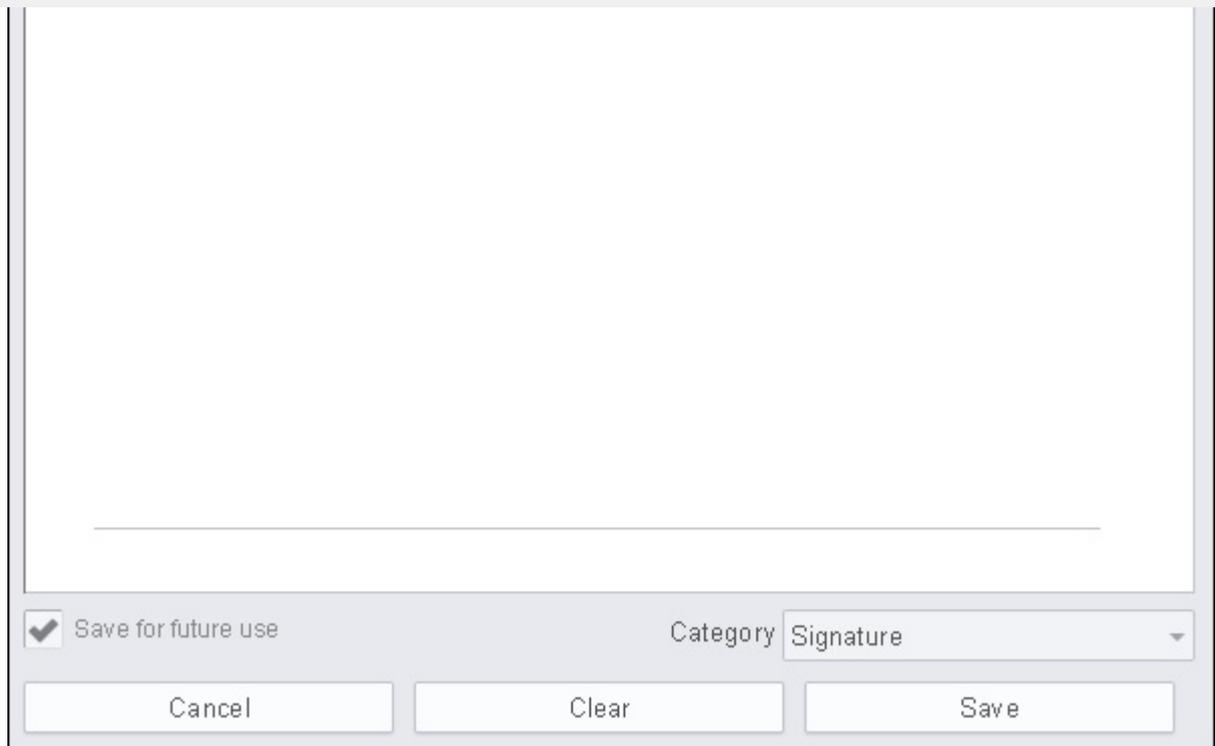
The following example shows the e-signatures displayed in the Viewer:



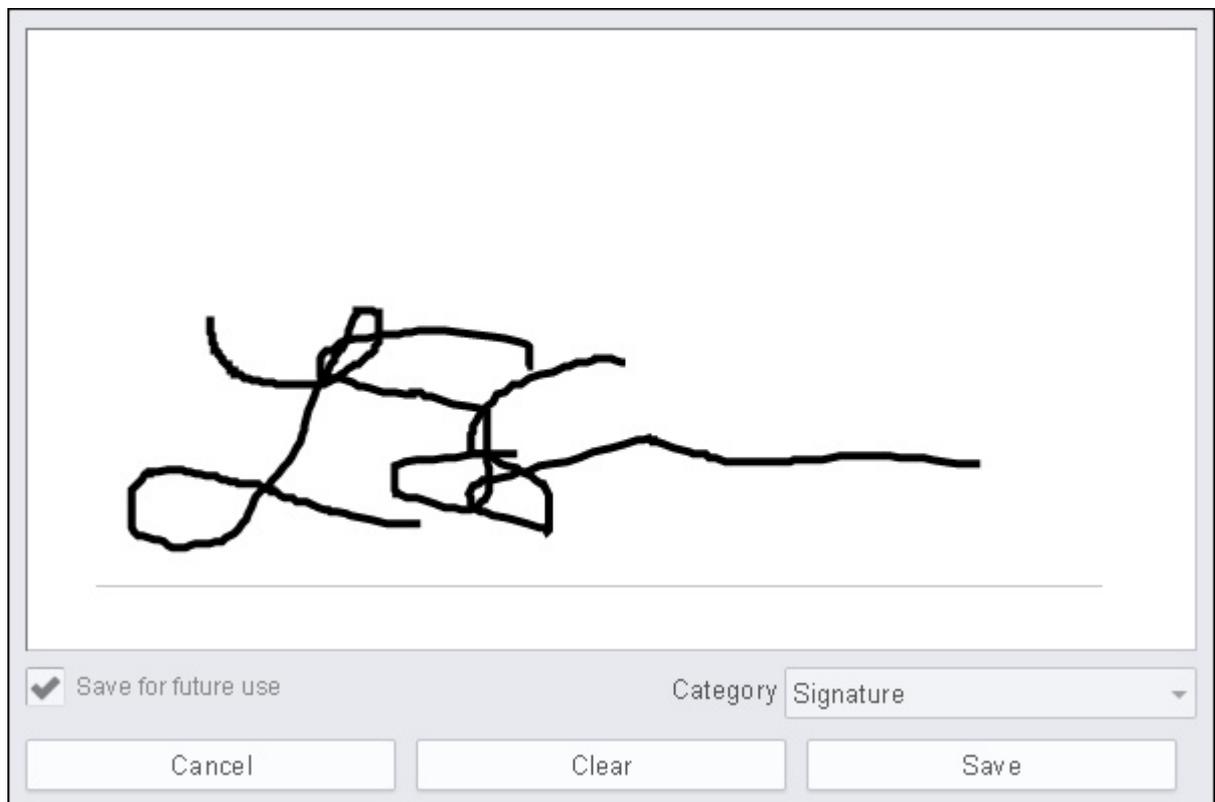
The following section covers how to create freehand and text signatures.

### To create a Freehand signature:

1. Click on the **Create Freehand Signature** icon. A dialog box displays:



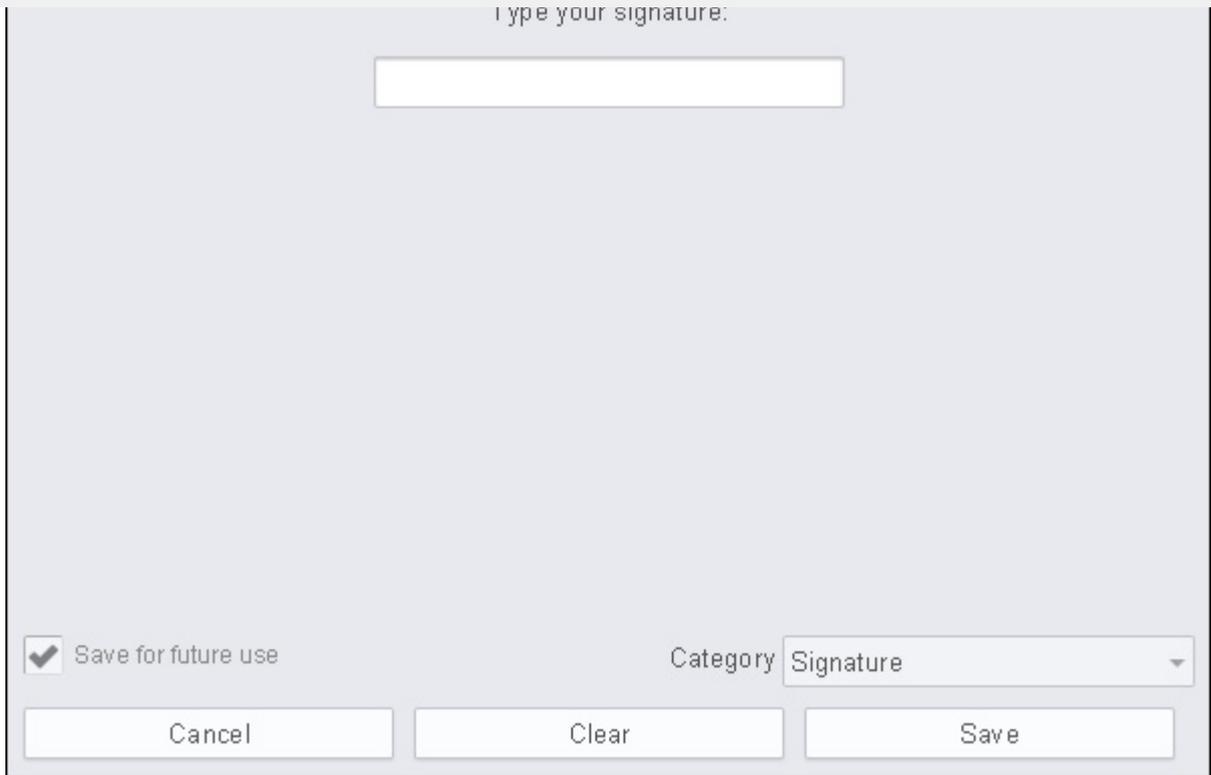
2. Using your mouse, start to draw your signature in the box:



3. Select the **Category**: Signature, Initials or Title and then click **Save**.

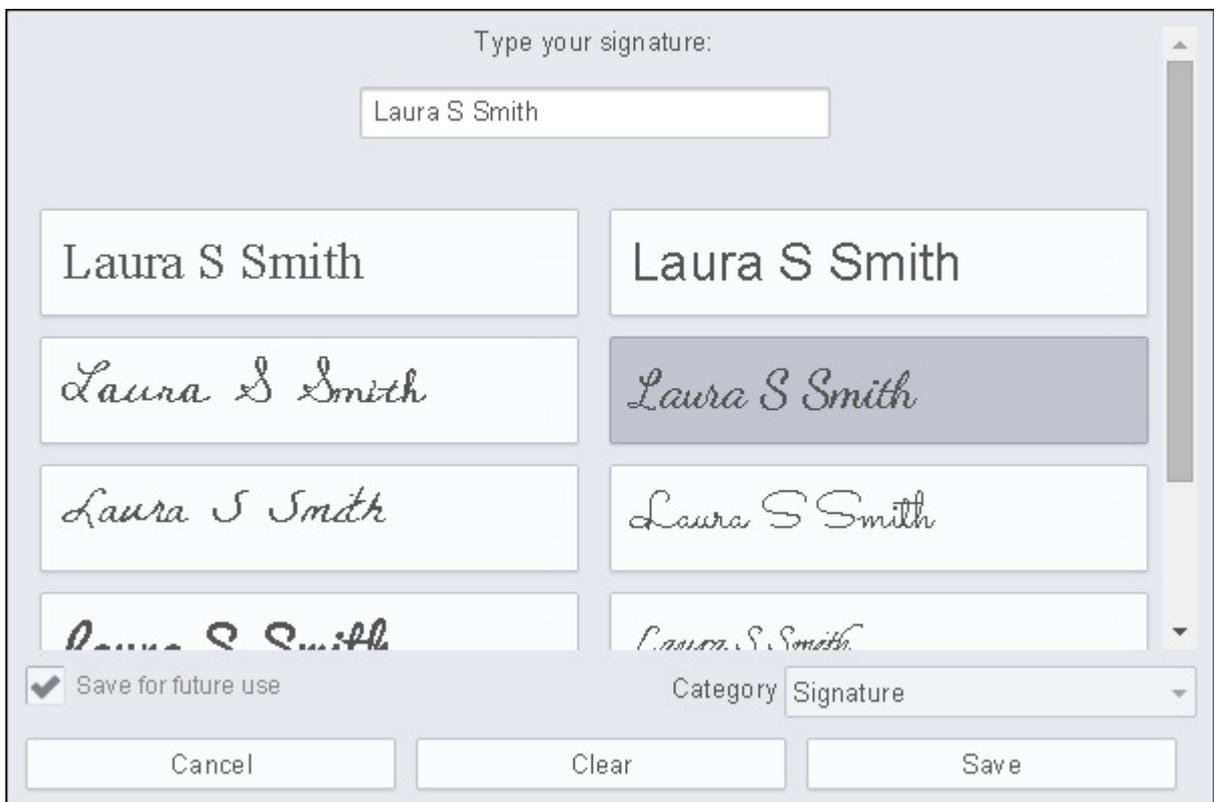
### To create a Text signature:

1. Click on the **Create Text Signature** icon. A dialog box displays:



The screenshot shows a dialog box titled "Type your signature:". At the top, there is a text input field that is currently empty. Below the input field, there is a checkbox labeled "Save for future use" which is checked. To the right of the checkbox is a dropdown menu labeled "Category" with "Signature" selected. At the bottom of the dialog, there are three buttons: "Cancel", "Clear", and "Save".

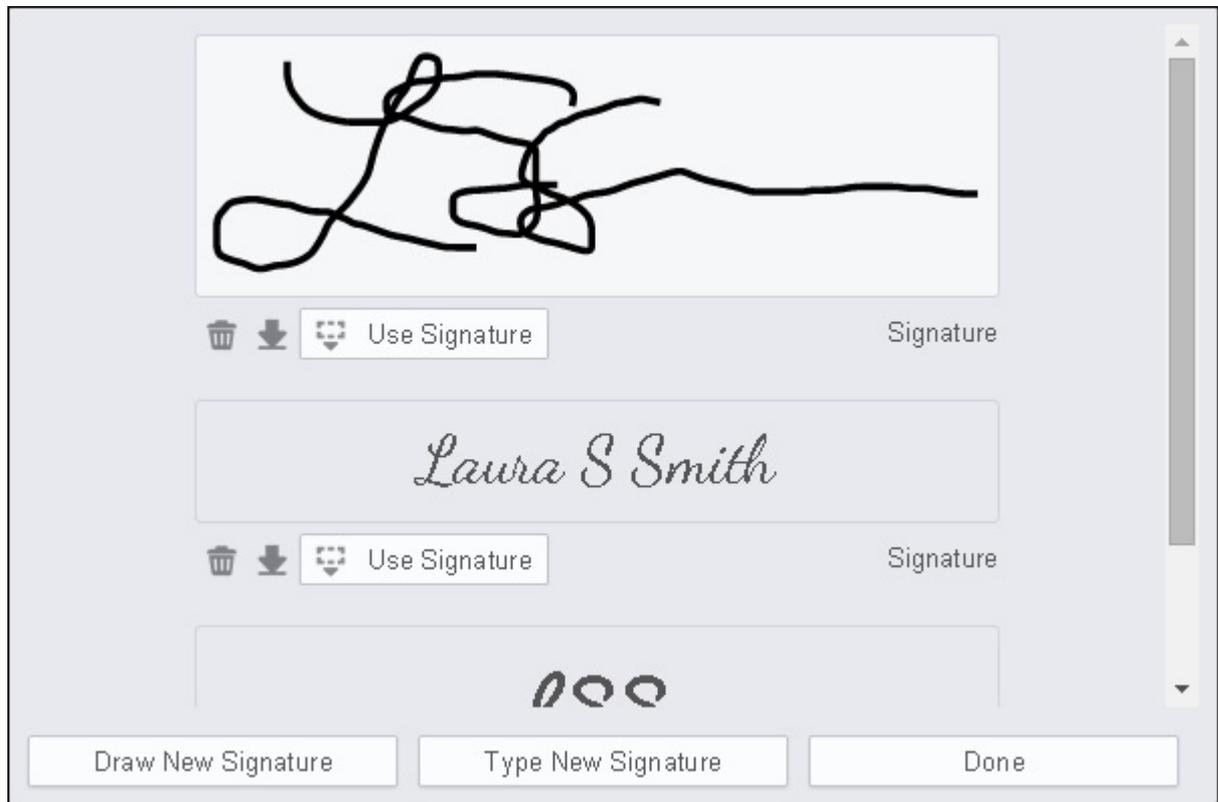
2. In the **Type Your Signature** box, enter your signature. Choose one of the pre-filled signatures with the font you prefer:



The screenshot shows the same dialog box as above, but now the text input field contains "Laura S Smith". Below the input field, there is a grid of eight buttons, each displaying the name "Laura S Smith" in a different font style. The second button in the second row, which uses a cursive font, is highlighted with a grey background. Below the grid, there is a checkbox labeled "Save for future use" which is checked. To the right of the checkbox is a dropdown menu labeled "Category" with "Signature" selected. At the bottom of the dialog, there are three buttons: "Cancel", "Clear", and "Save".

3. Select the **Category**: Signature, Initials or Title and then click **Save**.

1. Click on the **Manage E-Signatures** icon. A dialog box displays:



2. Click on the **Download Signature** button.
3. A signature.json file is downloaded to your computer. Click **Done**.

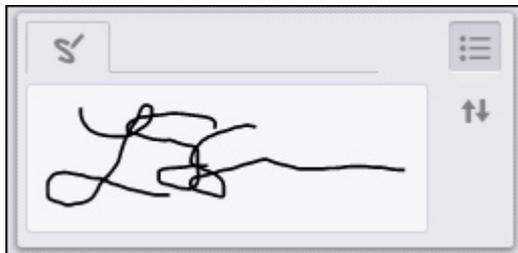
## Adding a Signature to a Document

To add a signature to a document:

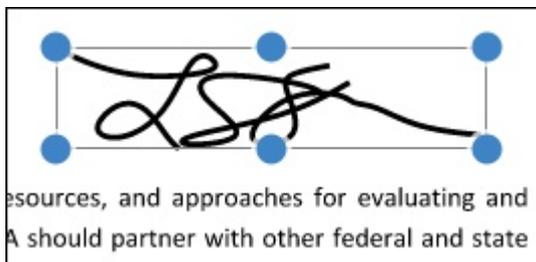
1. Click on the **Manage E-Signatures** icon. A dialog box displays:



2. Click on **Use Signature**. A floating toolbar displays in the Viewer:

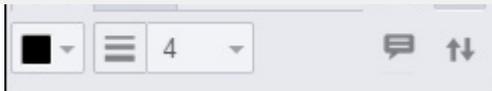


3. Click anywhere on the document where you want to place your signature. You can resize the signature using the yellow guide markers:



 If you have more than one signature on the document, click on the signature and the yellow guide markers appear so you can see which signature is active.

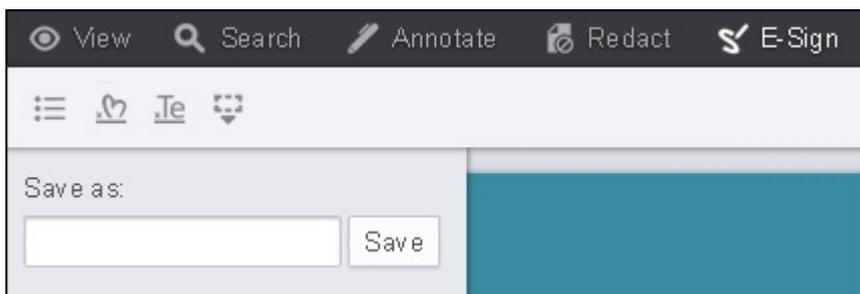
4. If you want to change the color, size, layer order of the signature or add a comment, you can select the attributes from the icons in the floating toolbar:



When in Horizontal Layout Mode, comments are not supported in the Viewer.

If you want to delete the signature, click on the **Trash can** icon.

5. To save the signature, click the **Save** icon. The **Save As** field displays on the left-hand side of the Viewer:



6. Enter a name for the signature and click **Save**.

## Creating Signature Categories

The signature categories are a configuration option for the jQuery plugin. In the `jQuery.fn` namespace, there is a property called `signatureCategories`. The default samples have this set to "Signature,Initials,Title", but if you want to change it or add your own types, you can edit the initialization parameter. When this parameter is set, there will be a dropdown to select the category (when creating a new Freehand or Text signature). This dropdown will not be present if the `signatureCategories` parameter is not set.

## How Signatures are Saved

Signatures are persisted in the browser's Local Storage so an end user can use the same signatures to sign across multiple documents and multiple sessions within the same browser. All signatures are stored in a JSON format under the storage key "pccvSignatures".

This storage method is a convenient way for end users to reuse previously created signatures, and is protected at the website domain level by the browser. However, this method may not be suitable for all business needs. The saving process is done inside `viewer.js`, in the E-Signature module, and can be overloaded, modified, or removed to match your business needs and criteria.

The Viewer expects signatures to be available in a collection object at `PCCViewer.Signatures`. Managing this collection is implemented as a module in the `viewer.js` file. See [Modifying viewer.js](#) for more information.

End users are also able to save their individual signatures for their own record using the "Download Signature" button under the Manage E-Signatures menu. This will download a plain-text JSON file of the selected signature.

## Key Board Shortcuts

The following keyboard shortcuts are supported in the Viewer:

### Viewer Shortcuts

Key	Action
Page Up	Previous Page
Page Down	Next Page
Ctrl + G	Go to page - The cursor moves to the page number to edit (in the upper right corner of the Viewer), the page number is highlighted, you enter the new page number and press Enter.
Arrow Keys	Move viewport - The Arrow Keys are the same as using the Scroll Bars.
+	Zoom In
-	Zoom Out
Del	Delete the currently selected annotation(s)/redactions(s).

### Dialog Box Shortcuts

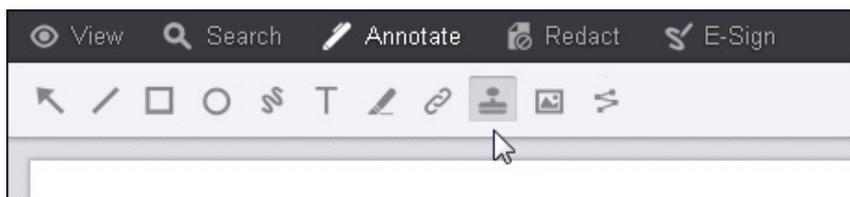
Key	Action
Esc	Cancel / Close

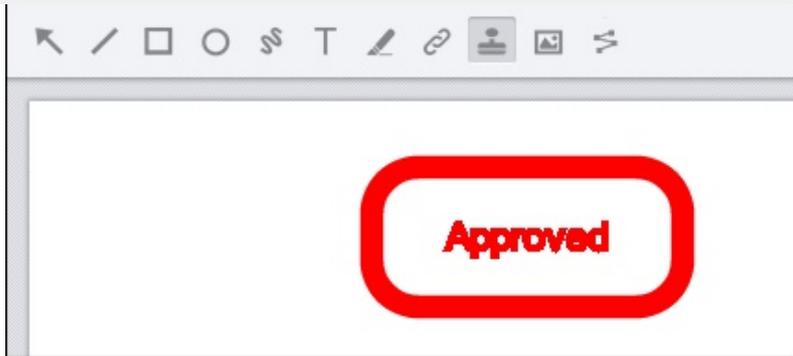
## Work with Sticky Mouse Tools

The mouse tools on the Annotation, Redaction and E-Signature menus can be selected to be used once or multiple times. The following section describes how to use this functionality in the Viewer. For API documentation on sticky mouse tools, refer to the [fn namespace](#) topic.

### To use the mouse tool once:

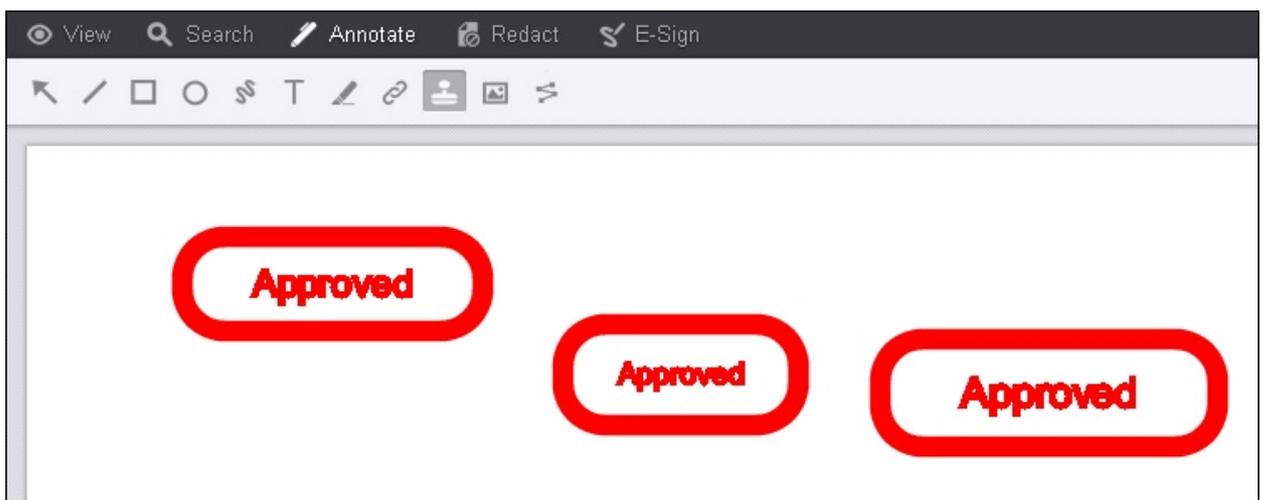
1. Click on the **mouse tool** once and you can use the mouse tool one time:





To use the mouse tool multiple times (known as "sticky mouse tool"):

1. Click on the **mouse tool** twice and it will lock the mouse tool so that you can use it multiple times. Note that when the mouse tool is clicked on twice, the background turns gray and the mouse tool turns white:



Further clicks on the mouse tool will toggle it from locked to unlocked.

 When you lock the mouse tool, it will disable touch-based scrolling.

2. To turn off the mouse tool completely, switch to a different tool. For example, click on the Hand Pan tool at the bottom of the Viewer:



 The default mouse tool is the Hand Pan tool.

## E-Signature Module

This section contains the following information:

- [E-Signing Overview](#)
- [Creators - Design a Template](#)
  - [Tools Reference](#)
  - [Set up a Basic Template](#)
  - [Add Features to the Template](#)
  - [Use the Form Field Detector](#)
    - [Form Field Detector Error Messages](#)
- [Signers - Fill out a Form](#)
  - [Tools Reference](#)
  - [Fill Out a Form](#)
  - [Sign a Form with Multiple Roles](#)
  - [Sign a Form Converted by the Form Field Detector](#)

## E-Signing Overview

The E-Signature module opens up a set of PrizmDoc workflows based on forms stored in any of the many [file formats supported](#) by PrizmDoc. As with the PrizmDoc Viewer, the E-Signature module works in your website to empower your customers as you see fit.

With v12.0, the introduction of forms detection makes the process faster. The E-signature module will [automatically detect](#) form fields in a PDF or raster image document and convert them so that it's easier to fill out your templates. Depending on the quality of the forms in the document, there may be no manual effort required by the [template designer](#).

The process starts with any document that you would like to "fill in" with some content. Note that this doesn't have to be an actual "form", it can really be any document that PrizmDoc can display.

If the document that you upload to the E-Signature module doesn't have any detectable form fields, a template designer then annotates it with the proper fields that will allow an [end user](#) to fill it out, burn in their signature, and create a finalized document that can then be forwarded as needed.

As an example, let's consider a blank Word document, a form for requesting time off:

**Absence Request**

**Absence Information**

Employee Name: \_\_\_\_\_  
Date: \_\_\_\_\_ Department: \_\_\_\_\_  
Manager: \_\_\_\_\_

Type of Absence Requested:

<input type="checkbox"/> Sick	<input type="checkbox"/> Vacation	<input type="checkbox"/> Bereavement	<input type="checkbox"/> Time Off Without Pay
<input type="checkbox"/> Other	<input type="checkbox"/> Jury Duty	<input type="checkbox"/> Maternity/Paternity	<input type="checkbox"/> Floating Holiday

Dates of Absence:  
From: \_\_\_\_\_ To: \_\_\_\_\_

Number of Hours Requesting \_\_\_\_\_ Number of Available Hours \_\_\_\_\_

Reason for Absence:  
\_\_\_\_\_

*You must submit requests for absences, other than sick leave, two weeks prior to the first day you will be absent.*

\_\_\_\_\_  
Employee Signature

\_\_\_\_\_  
Date

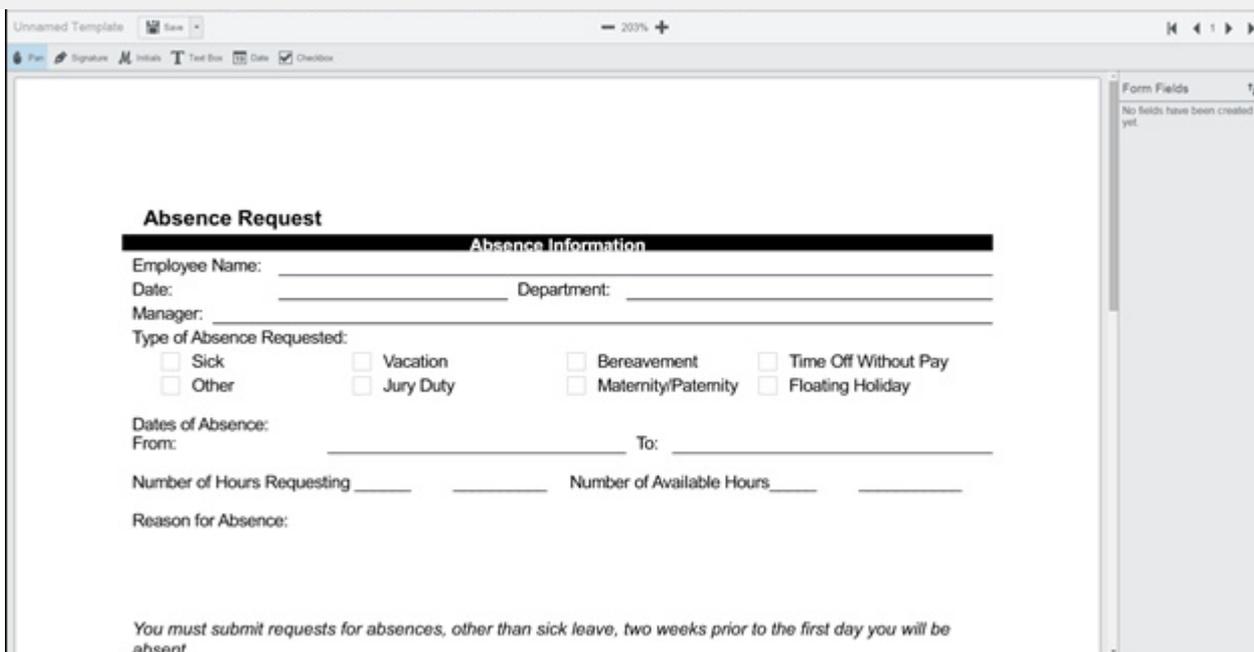
Normally, an employee would fill this out in Word, print it, sign and date it, then submit to their manager. This is not a very friendly workflow, and it requires both a copy of Word (expensive) and printing a physical copy of the form (wasteful).

The goal of our eSigning module is to make this workflow entirely electronic, including signing and producing a final electronic document that can be forwarded for approval. In order to make this workflow a reality, you would need to do the following:

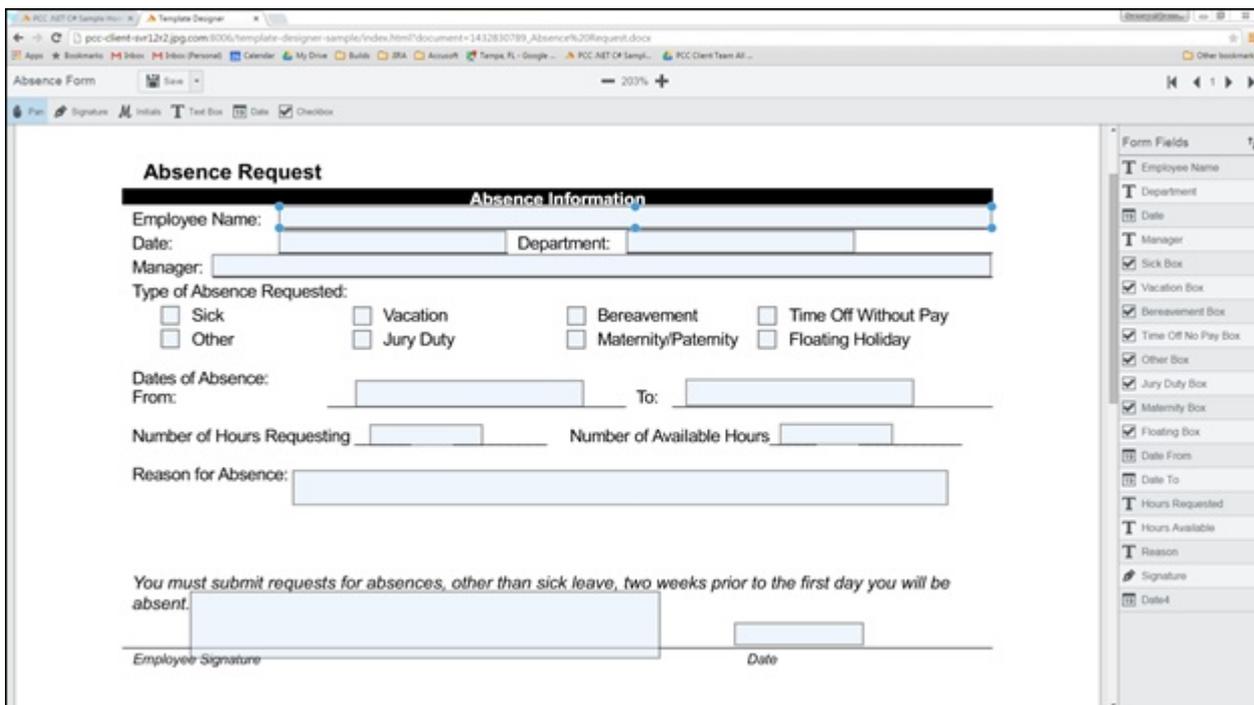
1. Set up a web-based workflow that incorporates our "Template Creator" module. This module allows creating a "template" out of a blank document by overlaying pre-defined form fields, such as Text, Date, or Signature fields. This template is then stored within the website for later use by employees wanting to fill out the form.
2. Set up a web-based workflow that incorporates our "Template Signer" module. This module allows opening a previously created template and then filling out the form's fields, including adding signatures and initials.

It's important to note that these two workflows are entirely independent. The people who might be template creators would normally be a very different set of people from those who need to fill out a form.

To illustrate the process, let's take the Absence Request form from above and load it in the template creator module:



Note the tools available in the upper left of the screen. Currently we support Signatures, Initials, Text, Dates, and Checkboxes. So let's create the template using these field types, shown below:



Here you can see the form fields that have been overlaid on the original document. The right hand panel shows the list of all fields by name and type. Once this template is created and saved, it's now ready to be consumed and filled in by users. Below you can see how this looks in the default template signing module:

**Absence Request**

**Absence Information**

Employee Name:  \*

Date:  \* Department:  \*

Manager:  \*

Type of Absence Requested:

Sick  Vacation  Bereavement  Time Off Without Pay

Other  Jury Duty  Maternity/Paternity  Floating Holiday

Dates of Absence:

From:  \* To:  \*

Number of Hours Requesting  \* Number of Available Hours  \*

Reason for Absence:

Progress: 0% [Download Signed Form](#)

Here you can see the field layout that users will see. The fields with red stars are the required fields and the progress bar at the bottom of the screen illustrates how complete the required fields are in the form. Once all required fields have been completed, the "Download Signed Form" button will be enabled, allowing the user to download a PDF of the fully completed form as shown below:

**Absence Request**

**Absence Information**

Employee Name:

Date:  Department:

Manager:

Type of Absence Requested:

Sick  Vacation  Bereavement  Time Off Without Pay

Other  Jury Duty  Maternity/Paternity  Floating Holiday

Dates of Absence:

From:  To:

Number of Hours Requesting  Number of Available Hours

Reason for Absence:

*You must submit requests for absences, other than sick leave, two weeks prior to the first day you will be absent.*

Employee Signature:  Date:

Progress: 100% [Download Signed Form](#)

Now that the form is complete, the download button will allow downloading the final burned document, shown below:

**Absence Request**

**Absence Information**

Employee Name: Jerry Smith  
Date: 05/28/2015 Department: Logistics  
Manager: Madeleine Wallis

Type of Absence Requested:

<input type="checkbox"/> Sick	<input checked="" type="checkbox"/> Vacation	<input type="checkbox"/> Bereavement	<input type="checkbox"/> Time Off Without Pay
<input type="checkbox"/> Other	<input type="checkbox"/> Jury Duty	<input type="checkbox"/> Maternity/Paternity	<input type="checkbox"/> Floating Holiday

Dates of Absence:  
From: 06/08/2015 To: 06/12/2015

Number of Hours Requesting 40 Number of Available Hours 73

Reason for Absence:

*You must submit requests for absences, other than sick leave, two weeks prior to the first day you will be absent.*

Jerry Smith 05/28/2015  
Employee Signature Date

This is just a quick overview that should illustrate how useful this could be in your product or your business. The most powerful part of the whole system is that it exists in your web pages, under your full control. And just like our Viewer, this viewer is heavily configurable via [APIs and eventing](#).

## Creators - Design a Template

This section contains information on how to design templates for end users:

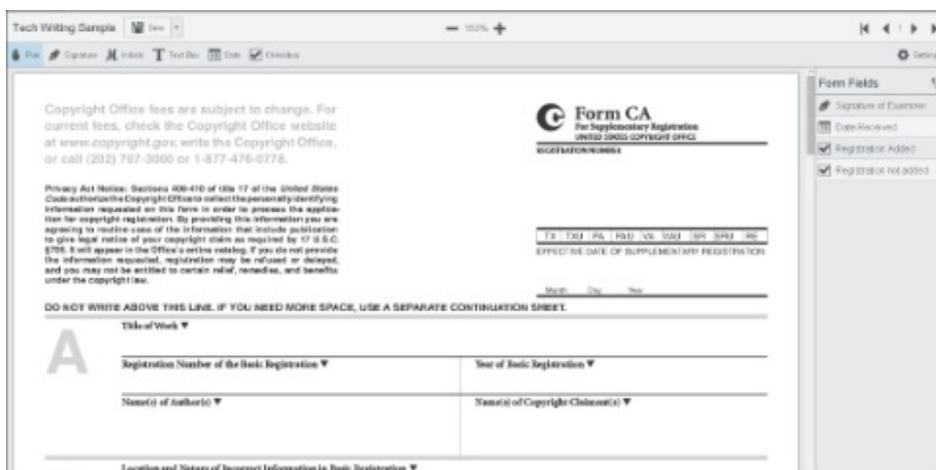
- [Tools Reference](#)
- [Set up a Basic Template](#)
- [Add Features to the Template](#)
- [Use the Form Field Detector](#)
  - [Form Field Detector Error Messages](#)

## Tools Reference

This section covers the following features:

- [Example of a Template in the E-signature Viewer](#)
- [Main Toolbar](#)
- [Adjusting a Field's Position](#)
- [Icons, Buttons & Fields](#)
- [Form Navigator](#)

The following example shows a template displayed in the main e-signature viewer:



## Main Toolbar

The following example shows the Main Toolbar:



## Pan

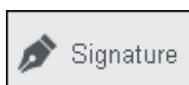
The following example shows the Pan icon:



Use the Pan feature to move the document up or down as desired.

## Signature

The following example shows the Signature icon:



When you click on the Signature button, the following items are displayed:

- Required Button - select the button to require the end user to fill in the field
- Read-only Button - select the button to set the field to be read-only
- Field ID - this field is populated with a generic name by default
- Display Name - enter a name to easily recognize the field

## Initials

The following example shows the Initials icon:

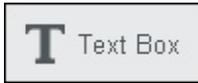


When you click on the Initials button, the following items are displayed:

- Read-only Button - select the button to set the field to be read-only
- Field ID - this field is populated with a generic name by default
- Display Name - enter a name to easily recognize the field

## Text Box

The following example shows the Text Box icon:

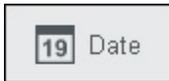


When you click on the Text Box button, the following items are displayed:

- Required Button - select the checkbox to require the end user to fill in the field
- Read-only Button - select the button to set the field to be read-only
- Multiline Checkbox - select the checkbox to allow the end user to enter multiple lines of text in the text box (the Max Font Size drop-down will display when you select this checkbox so you can choose the maximum font size you want the field to auto size to)
- Field ID - this field is populated with a generic name by default
- Display Name - enter a name to easily recognize the field
- Max Font Size Drop-down - when multiline is selected, the Max Font Size drop-down is displayed so you can select the maximum font size you would like the field to auto size to
- Color Drop-down - click to select from a list of available colors
- Character Limit - enter a number to limit the number of characters an end user can enter

## Date

The following example shows the Date icon:



When you click on the Date button, the following items are displayed:

- Required Button - select the button to require the end user to fill in the field
- Read-only Button - select the button to set the field to be read-only
- Field ID - this field is populated with a generic name by default
- Display Name - enter a name to easily recognize the field
- Font Drop-down - click to select from a list of available fonts
- Color Drop-down - click to select from a list of available colors

## Checkbox

The following example shows the Checkbox icon:



When you click on the Checkbox button, the following items are displayed:

- Required Button - select the button to require the end user to fill in the field
- Read-only Button - select the button to set the field to be read-only
- Field ID - this field is populated with a generic name by default

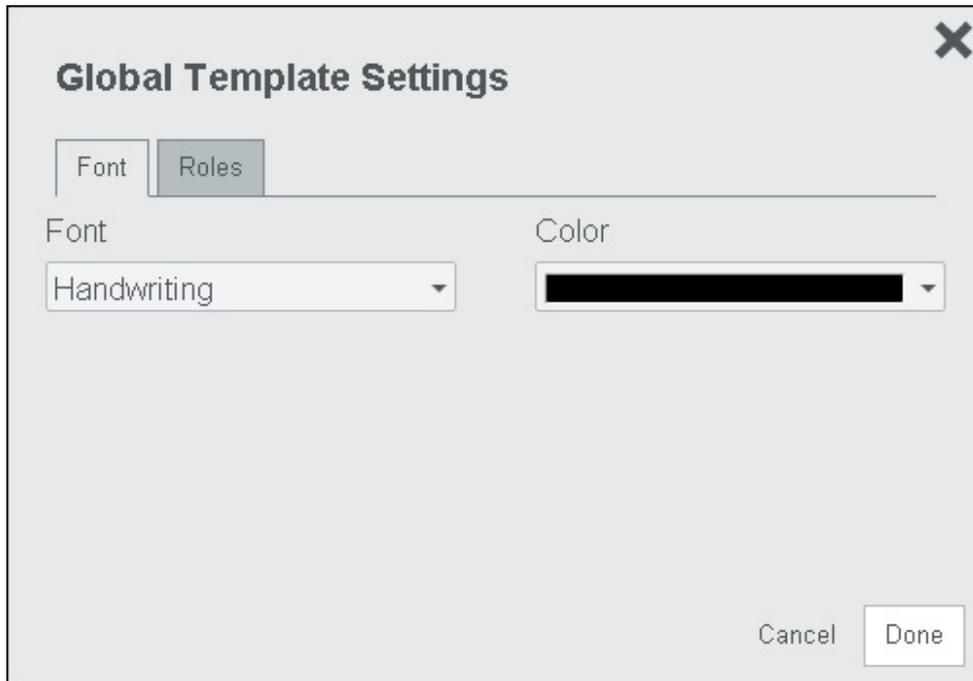
- Checkbox Group - enter a name for a group of checkboxes
- Allow Multiple Selections - select the checkbox to allow the end user to select multiple checkboxes in a group

## Settings

The following example shows the Settings icon:



1. To change the font or font color, click on **Settings** and the Global Template Settings dialog box displays:



### Font Tab

- Font Drop-down - click to select from a list of available fonts
- Color Drop-down - click to select from a list of available colors

### Roles Tab

- Role Name - type in the role name
- Color - click to select from a list of available colors
- Roles - displays the roles that have been created (maximum, five roles)



Note that when you select a font or color in the Global Template Settings dialog box, it is the default setting for the Text Box and Date fields that you create in the Template. You can override the font or font color on a per field basis as shown below.

## Changing Settings in the Properties Panel

1. When you create a Text Box and view the Properties Panel associated with that Text Box, you will see "Use Global Setting" as the default setting:



2. If you want to change the default setting for this Text Box only, select a different **font** or **font color** from the drop-down list here in the Properties Panel.

## Adjusting a Field's Position

### Arrow Keys

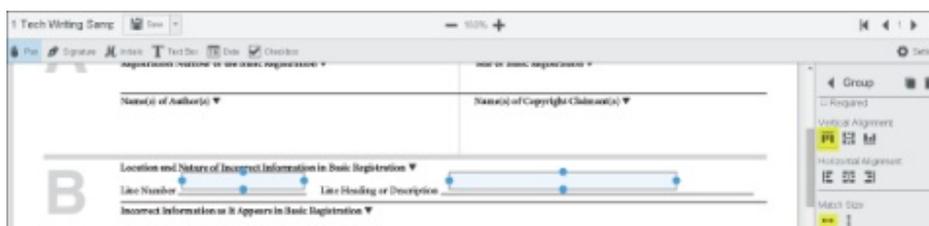
You can move or 'nudge' the position of a field by using your keyboard's arrow keys. Click on the **field** you want to move, then select the **up, down, left** or **right arrow key** on the keyboard to move the field to the desired location on the form.

## Group Properties Panel

1. You can align and match the size of multiple fields at one time by using the **Ctrl** key and clicking on the fields you want to align:



2. Then select **Match Size** and **Vertical Alignment** as shown in yellow highlight below:



 You can align and match the size for the signature, initials, text box, and date fields.

## Icons, Buttons & Fields

### Unnamed Template Field



- Unnamed Template Field - click to enter the name of the template you are designing

### Save Button

The following example shows the Save Button / Drop-down:



- Save Drop-down - click to Save or Save a Copy of the template:



### Zoom In / Zoom Out

The following example shows the Zoom In / Zoom Out Toolbar:



- Minus Sign - click to zoom out
- Percentage - displays the viewing percentage
- Plus Sign - click to zoom in

### Page Navigation

The following example shows the Page Navigation Toolbar:



- First Page - click to go to the first page of the template
- Previous Page - click to go to the previous page
- Next Page - click to go to the next page
- Last Page - click to go to the last page of the template

### Required Fields

The following example shows the red indicator for a required field:

DO NOT WRITE ABOVE THIS LINE. IF YOU NEED MORE SPACE, USE A SEPARATE C

A	<b>Title of Work ▼</b> <input type="text"/>
	<b>Registration Number of the Basic Registration ▼</b> <input type="text"/>
	<b>Name(s) of Author(s) ▼</b> <input type="text"/>

Once you click on the required field, the indicator will disappear. When you click outside of the field, the red indicator will reappear.

### Read-Only Fields

The following example shows the indicator for a read-only field:

Date:	<input type="text"/>
-------	----------------------

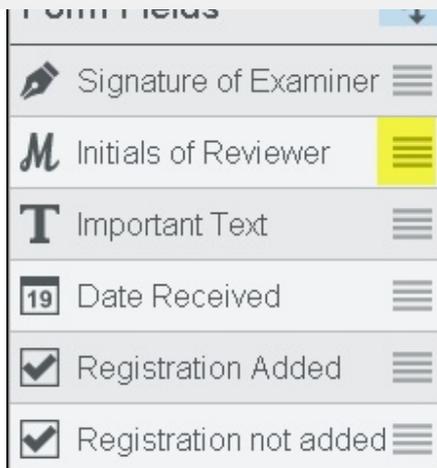
Once you click on the read-only field, the indicator will disappear. When you click outside of the field, the red indicator will reappear.

### Form Navigator

1. After you add some fields to the template, you will see them displayed in the Form Navigator to the right of the template:

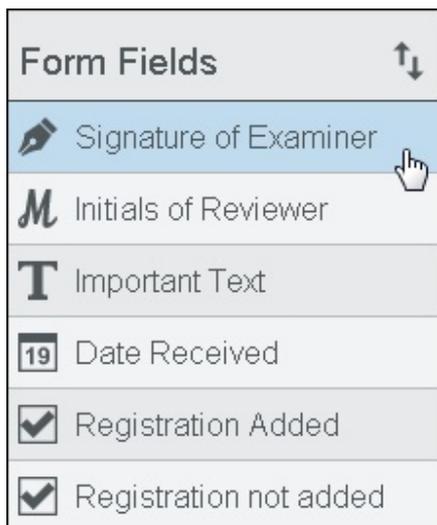
Form Fields <span style="float: right;">↕</span>	
	Signature of Examiner
	Initials of Reviewer
	Important Text
	Date Received
<input checked="" type="checkbox"/>	Registration Added
<input checked="" type="checkbox"/>	Registration not added

2. You can sort the Form Field names so the proper tab order is created for signing. Click on the **arrows** (highlighted in blue below) to activate the sorting feature. Click on the **bars** (highlighted in yellow below) to drag the Form Field name up or down in the list:

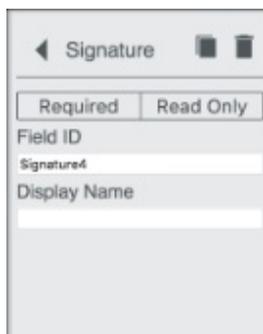


After you have moved the fields as desired, click on the **arrows** again to de-activate the sorting feature.

3. You can easily modify the fields from this pane by clicking on the Display Name of the field. For example, click on **Signature of Examiner** to edit that field:



4. The Signature Pane displays and you can edit the contents of this pane as desired. The **Duplicate** icon, highlighted in yellow below, allows you to draw multiple signatures at a time. To delete a signature field in the template, click on the **Delete** button, highlighted in green below:



### Properties Panel

The following example shows the Properties Panel for a Text Box (which displays when you click on an item in the

The screenshot shows the Properties Panel for an 'Address' field. At the top, there is a title bar with a back arrow, the text 'Address', and two trash icons. Below this are two rows of buttons: 'Required' and 'Read Only' in the first row, and 'Multiline' (highlighted in blue) and 'Single Line' in the second row. The 'Field ID' section contains a text input field with 'Text1'. The 'Field Type' section has a dropdown menu set to 'Text Box'. The 'Alignment' section features three icons representing left, center, and right alignment, with the left alignment icon highlighted in blue. The 'Font' section has a dropdown menu set to 'Use Global Setting'. The 'Color' section also has a dropdown menu set to 'Use Global Setting'. The 'Max Font Size' section has a dropdown menu set to '8 pt'. The 'Character Limit' section has a text input field containing '1000'.

Note that the following items are available in the Properties Panel depending on the tool you selected on the Main toolbar (Signature, Initials, Text Box, Date, Checkbox):

- Required Button - select the button to require the end user to fill in the field
- Multiline Checkbox - select the checkbox to allow the end user to enter multiple lines of text in the text box (the Max Font Size drop-down will display when you select this checkbox so you can choose the maximum font size you want the field to auto size to)
- Read-only Button - select the button to set the field as read-only for the end user
- Field ID - this field is populated with a generic name by default
- Alignment - sets the horizontal text alignment for the selected field
- Display Name - enter a name to easily recognize the field
- Font Drop-down - the default is "Use Global Setting" or you can click to select from a list of available fonts
- Color Drop-down - the default is "Use Global Setting" or you can click to select from a list of available colors
- Max Font Size Drop-down - when Multiline is selected, the Max Font Size drop-down is displayed so you can select the maximum font size you would like the field to auto size to
- Character Limit - enter a number to limit the number of characters a user can enter

## Setup a Basic Template

This topic contains the steps to set up a template:

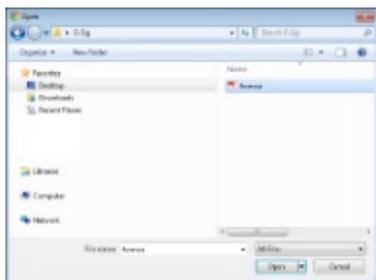
- [Creating a New Template](#)
- [Viewing Required Fields](#)
- [Viewing Read-only Fields](#)
- [Adjusting a Field's Horizontal Alignment](#)
- [Adjusting a Field's Position](#)
- [Sorting Fields](#)
- [Editing, Duplicating or Deleting a Field](#)
- [Saving the Template](#)
- [Deleting the Template](#)

### Creating a New Template

1. From the splash screen, under Create a New Template, add a **file** by selecting **one of the following**:
  - Drop a File Here
  - Click to Select a File
2. For this example, choose **Click to Select a File**:



3. The Open dialog box displays. Browse to the file location, click on the **file** and click **Open**:



4. The template displays in the E-signing window:

# PrizmDoc v12.3 - Updated June 23, 2017

565

Copyright Office fees are subject to change. For current fees, check the Copyright Office website at [www.copyright.gov](http://www.copyright.gov), write the Copyright Office, or call (202) 707-3000 or 1-877-476-0778.

**Form CA**  
For Supplementary Registration  
UNITED STATES COPYRIGHT OFFICE

Privacy Act Notice: Sections 406-410 of title 17 of the United States Code authorize the Copyright Office to collect the personally identifying information requested on this form in order to process the application for copyright registration. By providing this information you are agreeing to routine use of the information that includes publication to give legal notice of your copyright claim as required by 17 U.S.C. §705. It will appear in the Office's online catalog. If you do not provide the information requested, registration may be refused or delayed, and you may not be entitled to certain relief, remedies, and benefits under the copyright laws.

DO NOT WRITE ABOVE THIS LINE. IF YOU NEED MORE SPACE, USE A SEPARATE CONTINUATION SHEET.

Title of Work ▼

Registration Number of the Basic Registration ▼ Title of Basic Registration ▼

Name(s) of Author(s) ▼ Name(s) of Copyright Claimant(s) ▼

Location and Nature of Incorrect Information in Basic Registration ▼

Form Fields

- Signature of Examinee
- Date Received
- Registration Address
- Registration not added

5. In the upper left corner, click in the **Unnamed Template** field. Type in a **name** for the template and click **Save**:

Unnamed Template Save

Signature Initials Text Box Date 19 Checkbox

6. To make Global Settings for the template, click **Settings**:

Settings

7. The Global Template Settings dialog box displays:

**Global Template Settings**

Font Roles

Font Handwriting Color

Cancel Done

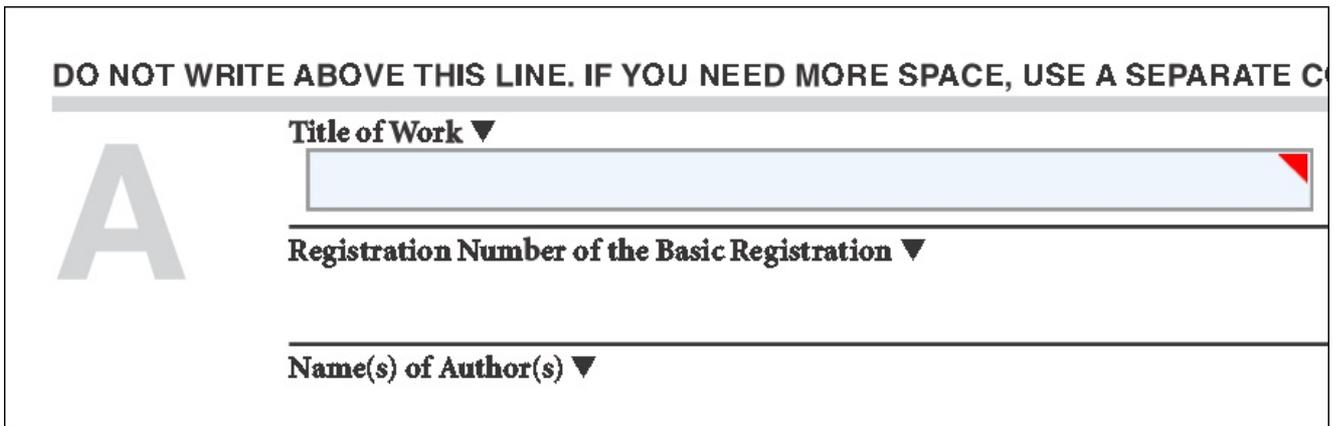
section below, "Adding Roles".

 Note that when you select a font or color in the Global Template Settings dialog box, it is the default setting for the Text Box and Date fields that you create in the Template. You can override the font or color on a per field basis if desired.

9. Once you are finished making selections in the Global Template Settings dialog box, click **Done**.

## Viewing Required Fields

Required fields can easily be seen by the red indicator in the upper right-hand corner:



Once you click on the required field, the indicator will disappear. When you click outside of the required field, the red indicator will reappear in the upper right-hand corner.

## Viewing Read-only Fields

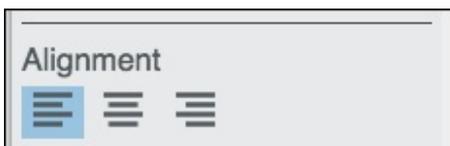
Read-only fields can be easily seen by the indicator in the upper right-hand corner:



Once you click on the read-only field, the indicator will disappear. When you click outside of the read-only field, the indicator will reappear in the upper right-hand corner.

## Adjusting a Field's Horizontal Alignment

You can specify a text and signature field's horizontal alignment by selecting one of the following buttons:

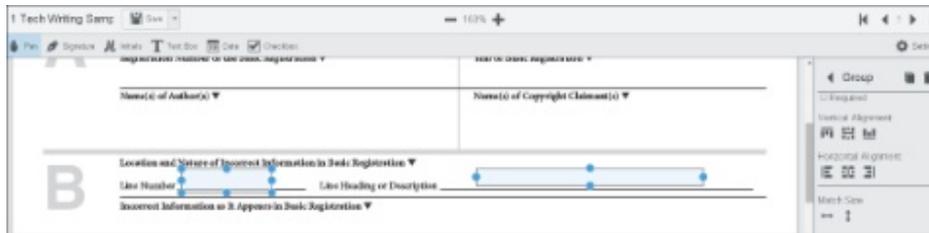


## Adjusting a Field's Position

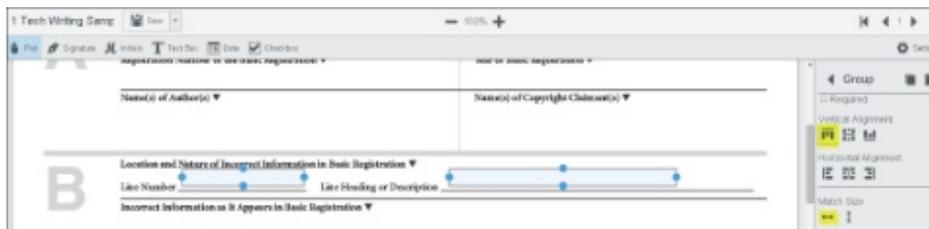
### Arrow Keys

You can move or 'nudge' the position of a field by using your keyboard's arrow keys. Click on the **field** you want to move, then select the **up**, **down**, **left** or **right arrow key** on the keyboard to move the field to the desired location on the form.

1. You can align and match the size of multiple fields at one time by using the **Ctrl key** and clicking on the fields you want to align:



2. Then select **Match Size** and **Vertical Alignment** as shown in yellow highlight below:



 You can align and match the size for the signature, initials, text box, and date fields.

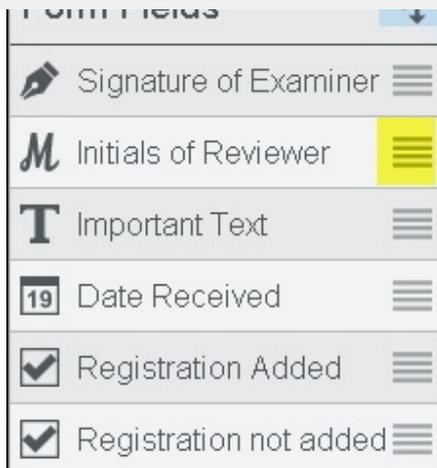
## Sorting Fields

Use the following steps to sort fields that you have created:

1. After you add some fields to the template, you will see them displayed in the Form Navigator to the right of the template:

Form Fields 	
	Signature of Examiner
	Initials of Reviewer
	Important Text
	Date Received
<input checked="" type="checkbox"/>	Registration Added
<input checked="" type="checkbox"/>	Registration not added

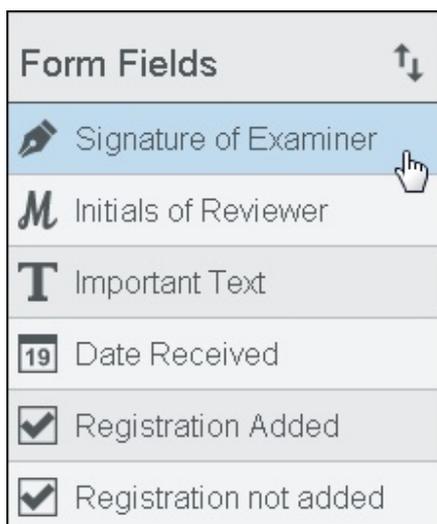
2. You can sort the Form Field names so the proper tab order is created for signing. Click on the **arrows** (highlighted in blue below) to activate the sorting feature. Click on the **bars** (highlighted in yellow below) to drag the Form Field name up or down in the list:



3. After you have moved the fields as desired, click on the **arrows** again to de-activate the sorting feature.

### Editing, Duplicating or Deleting a Field

1. You can easily modify the fields from the Form Navigator by clicking on the **Display Name** of the field. For example, click on **Signature of Examiner** to edit that field:



2. The Properties Panel displays and you can edit the contents of this pane as desired.

- To draw multiple signatures at a time, click on the **Duplicate** icon.
- To delete a signature field on the form, click on the **Delete** button.



3. When you are finished, click on the **Return** button to go back to the Form Navigator:



### Saving the Template

Use the following steps to save the template:

#### Save the Template

1. To save the template you created, click **Save** on the toolbar at the top of the Viewer window:



2. A check mark appears on the Save button to indicate your changes have been saved.

#### Save a Copy of the Template

1. To save a copy of the template you created with a different name, click the **Save drop-down** and select **Save a copy**:



2. A copy of the template is created. Click in the **Name Field** and enter the new template name:



3. Click **Save** to save the new template.

 If you exit a template without saving your changes, a message appears, "Confirm Navigation - Warning: You have unsaved changes, save before continuing. Are you sure you want to leave this page?" Click 'Leave this Page' to exit without saving changes or click 'Stay on this Page' to save your changes.

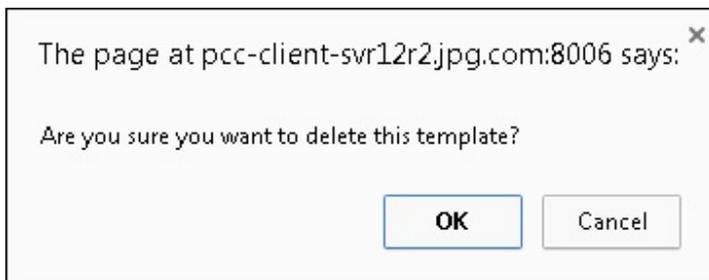
## Deleting the Template

Use the following steps to delete a template:

1. From the splash screen, under the heading 'View an Existing Template', you will see the list of available templates:



2. Scroll to the template you want to delete.
3. Click **Delete** and a message appears, "Are you sure you want to delete this template?"



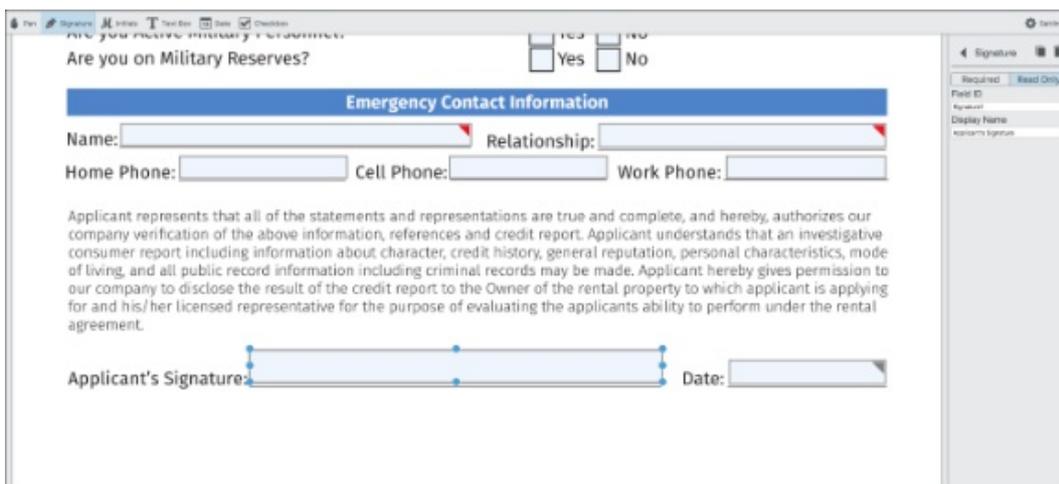
4. Select **OK** to delete the template (or Cancel to keep the template).

This topic contains the steps to design a template using all of the tools on the menu:

- Adding a Signature
- Adding Initials
- Adding a Text Box
- Adding a Date
- Adding a Checkbox
- Adding Roles
- Assigning Roles to Fields
- Assigning an Input Mask to a Text Box

## Adding a Signature

Use the following steps to add a signature to the template:



1. To add a signature, click on the **Signature** button.
2. Scroll to the location on the template you want to add the signature and draw a **box**.
3. The Properties Panel opens to the right of the template and displays the available options for the field.
4. If the signature is read-only, click the **Read-only** button.
5. If the signature is mandatory, click the **Required** button.
6. Enter a **Display Name** to easily refer to the form field. In this example, type in **Signature for Examiner**.

fields as desired.

You can adjust a field's position in several ways; see the section below, "Adjusting a Field's Position" for details.

## Adding Initials

Use the following steps to add initials to the template:

The screenshot shows the PrizmDoc interface with a tax form template. The 'Initials' properties panel is open on the right side, showing options for 'Required', 'Read Only', 'Field ID', and 'Display Name'. The form template is a 2014 Form 1040EZ, 'Income Tax Return for Single and Joint Filers With No Dependents'. The form fields are highlighted with blue boxes, indicating they are selected for editing. The 'Income' section is visible, with five lines of income information.

1. To add initials, click on the **Initials** button. Scroll to the location on the template you want to add the initials and draw a **box**.
2. The Properties Panel opens to the right of the template and displays the available options for the field.
3. If the initials are read-only, click the **Read-only** button.
4. If the initials are mandatory, click the **Required** button.
5. Enter a **Display Name** to easily refer to the form field. In this example, type in **Initials of Reviewer**.
6. If you want to add additional initials fields to the template, click on the **Duplicate** icon and draw the initials fields as desired.

You can adjust a field's position in several ways; see the section below, "Adjusting a Field's Position" for details.

## Adding a Text Box

Use the following steps to add a text box to the template:

The screenshot shows the PrizmDoc interface with the same tax form template. The 'Text Box' properties panel is open on the right side, showing options for 'Required', 'Read Only', 'Subline', 'Single Line', 'Field ID', 'Font', 'Display Name', 'Name Self', 'Color', and 'Character Limit'. The form template is the same as in the previous screenshot, but the 'Text Box' panel is now open, indicating that a text box has been added to the template.

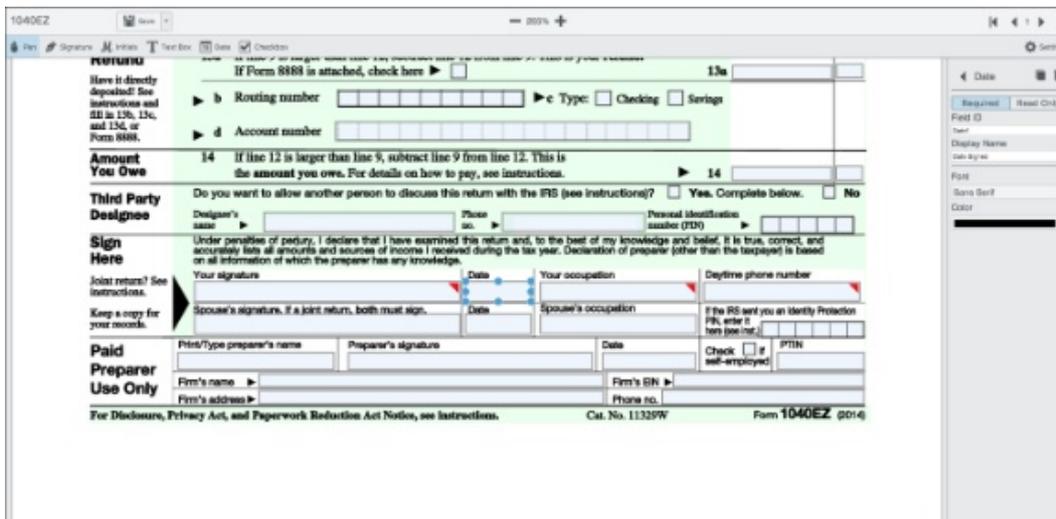
1. To add a text box, click on the **Text Box** button. Scroll to the location on the template you want to add the text

2. The Properties Panel opens to the right of the template and displays the available options for the field.
3. If the text box is read-only, click the **Read-only** button.
4. If the text box is mandatory, click the **Required** button.
5. If you want to allow the end user to enter multiple lines of text in the Text Box, click on the **Multiline** checkbox. Once you select the Multiline checkbox, the Font Size drop-down displays on the Properties Panel. Select the **font size** for the Multiline Text Box.
6. Enter a **Display Name** to easily refer to the form field. In this example, type in **Important Text**.
7. For a text box, you can choose a font type, a font color and you can limit the number of characters that are entered into the text box. Note that the font type and font color default selections are "Use Global Setting".
8. If you want to add additional text box fields to the template, click on the **Duplicate** icon and draw the text box fields as desired.

 You can adjust a field's position in several ways; see the section below, "Adjusting a Field's Position" for details.

## Adding a Date

Use the following steps to add a date to the template:



1. To add a date box, click on the **Date** button. Scroll to the location on the template you want to add the date and draw a **box**.
2. The Properties Panel opens to the right of the template and displays the available options for the field.
3. If the date box is read-only, click the **Read-only** button.
4. If the date box is mandatory, click the **Required** button.
5. Enter a **Display Name** to easily refer to the form field. In this example, type in **Date Received**.
6. For a date box, you can choose a font type and a font color. Note that the font type and font color default selections are "Use Global Setting".
7. If you want to add additional date fields to the template, click on the **Duplicate** icon and draw the date fields as desired.

 You can adjust a field's position in several ways; see the section below, "Adjusting a Field's Position" for details.

## Adding a Checkbox

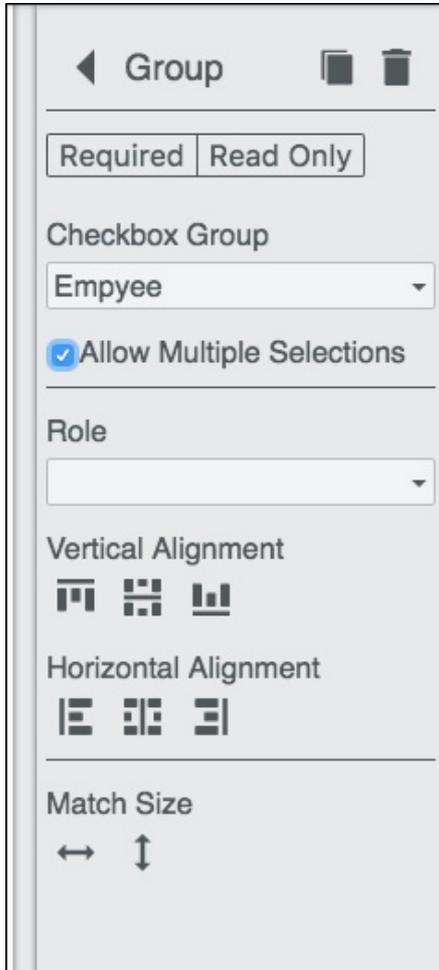
Use the following steps to add a checkbox to the template:

The image shows a screenshot of a tax form, specifically Form 1040EZ (2014). A green highlight is applied to the 'Amount You Owe' section, which includes a calculation instruction: '14 If line 12 is larger than line 9, subtract line 9 from line 12. This is the amount you owe. For details on how to pay, see instructions.' Below this, there are fields for 'Amount You Owe' and a checkbox for 'Do you want to allow another person to discuss this return with the IRS (see instructions)?'. The form also includes sections for 'Third Party Designee', 'Sign Here', and 'Paid Preparer Use Only'. A properties panel is visible on the right side of the form, showing options like 'Required', 'Read Only', 'Field ID', 'Display Name', 'Set member', 'Checkbox Group', and 'None'.

1. To add a checkbox, click on the **Checkbox** button. Scroll to the location on the template you want to add the checkbox and draw a **box**.
2. The Properties Panel opens to the right of the template and displays the available options for the field.
3. If the checkbox is read-only, click the **Read-only** button.
4. If the checkbox is mandatory, click the **Required** button. If not, leave it unchecked.
5. Enter a **Display Name** to easily refer to the form field. In this example, type in **Registration Added**.
6. If you want to add additional checkbox fields to the template, click on the **Duplicate** icon and draw the checkbox fields as desired.
7. To create a group of checkboxes, follow **step 5** above and create the desired amount of checkboxes that you want the user to choose from.
8. Press the **Ctrl key** and click on (multi-select) the checkboxes that you want to put into a group. The Group properties panel displays:

The image shows a screenshot of the 'Group' properties panel. At the top, there is a 'Group' title with a back arrow and two trash icons. Below the title, there are two buttons: 'Required' and 'Read Only'. The 'Checkbox Group' section has a dropdown menu currently set to 'None'. The 'Role' section has a dropdown menu. The 'Vertical Alignment' section has three icons representing different alignment options. The 'Horizontal Alignment' section has three icons representing different alignment options. The 'Match Size' section has two icons representing different match size options.

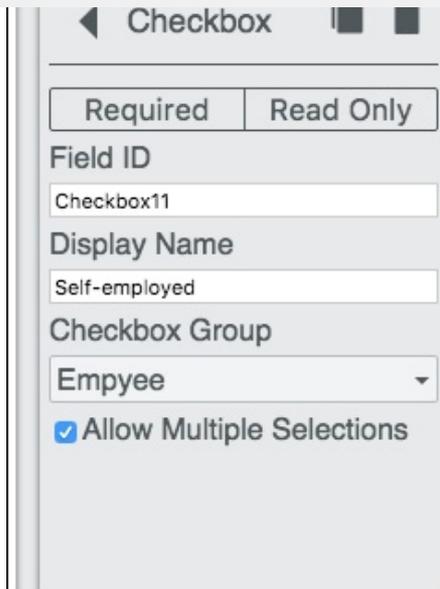
9. From the Group properties panel, you can select the **Required** button (if desired), the **Read Only** button (if desired), enter a **Checkbox Group** name, select the **alignment** and **match the size** of all the checkboxes in the group:



10. Once you enter a **Checkbox Group** name, you can select **Allow Multiple Selections** so that the user can check multiple checkboxes within that group.

 The group name is limited to 140 characters.

11. If you select **one checkbox** in the group, the properties panel will display the following:



12. From this example, you can see that the checkbox is required, part of a group (group name) and the group allows multiple selections. If you want to add another checkbox to the group, click on the **Duplicate** icon. To delete this specific checkbox from the group, click on the **Trash** icon.

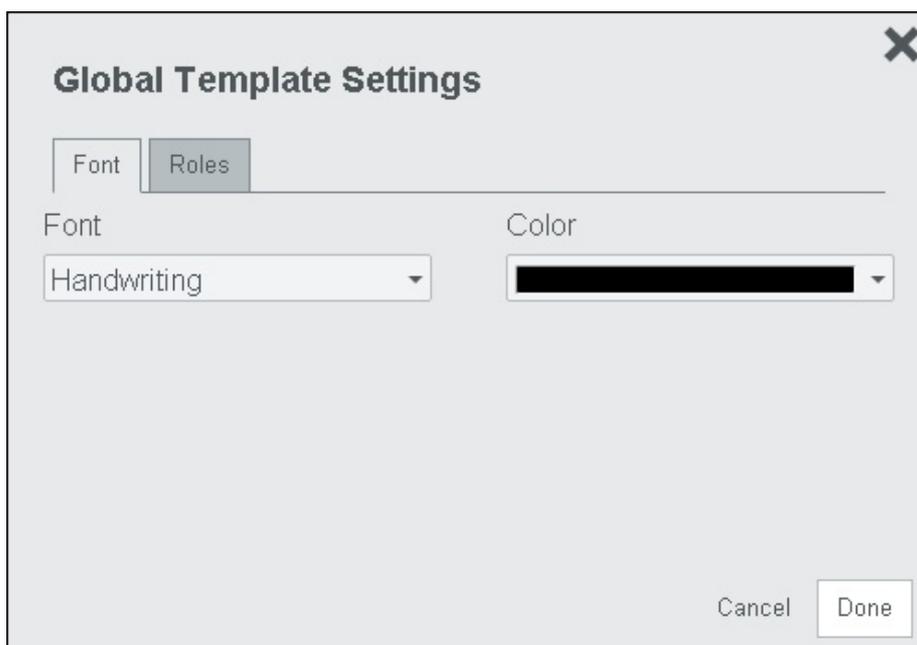
 You can adjust a field's position in several ways; see the section below, "Adjusting a Field's Position" for details.

## Adding Roles

You can add Roles to a template so that multiple people can review, edit, sign and save a document. For example, roles could be created for an Employee, a Manager and a Director. Adding roles to a template helps clarify the document review process in an organization by only allowing role-based interaction within the document.

Use the following steps to add roles to the template:

1. To add roles to a template, click on **Settings** and the Global Template Settings dialog is displayed:



2. Click the **Roles** tab and the roles settings are displayed:

**Global Template Settings**

Font Roles

Role Name

Roles  
*No roles have been created.*

Color

Cancel Done

3. In the **Role Name** field, type in **Employee** and click the **Checkmark** (or press Enter):

**Global Template Settings**

Font Roles

Role Name  ✓

Roles  
*No roles have been created.*

Color

Cancel Done

The new Role titled "Employee" is displayed under Roles in blue:

**Global Template Settings**

Font Roles

Role Name

Roles

Employee x

Color

Cancel Done

The colors apply to each role by default in the following order: blue, green, red, gold and black. If you want to select a different color for a role, you must select the color before clicking the "Checkmark" or pressing "Enter". Note the maximum number of roles you can create is five.

- Repeat steps 1-3 and create a Manager (color green) and Director (color red) role. When you are done creating roles, click **Done**:

**Global Template Settings**

Font Roles

Role Name

Roles

Employee x

Manager x

Director x

Color

Cancel Done

- Go to the top of the template and click **Save**.

### Assigning Roles to Fields

This section assumes you have created roles in the section above titled, "Adding Roles".

Use the following steps to assign roles to fields in the template:

- In this example, click on the **Form CA Received** field and the Properties Panel is displayed:

employer. You must also  
n if you do not get a

---

spouse if married  
e can claim you as a

A.

B.

C.

D.

E.

Group

Required Read Only

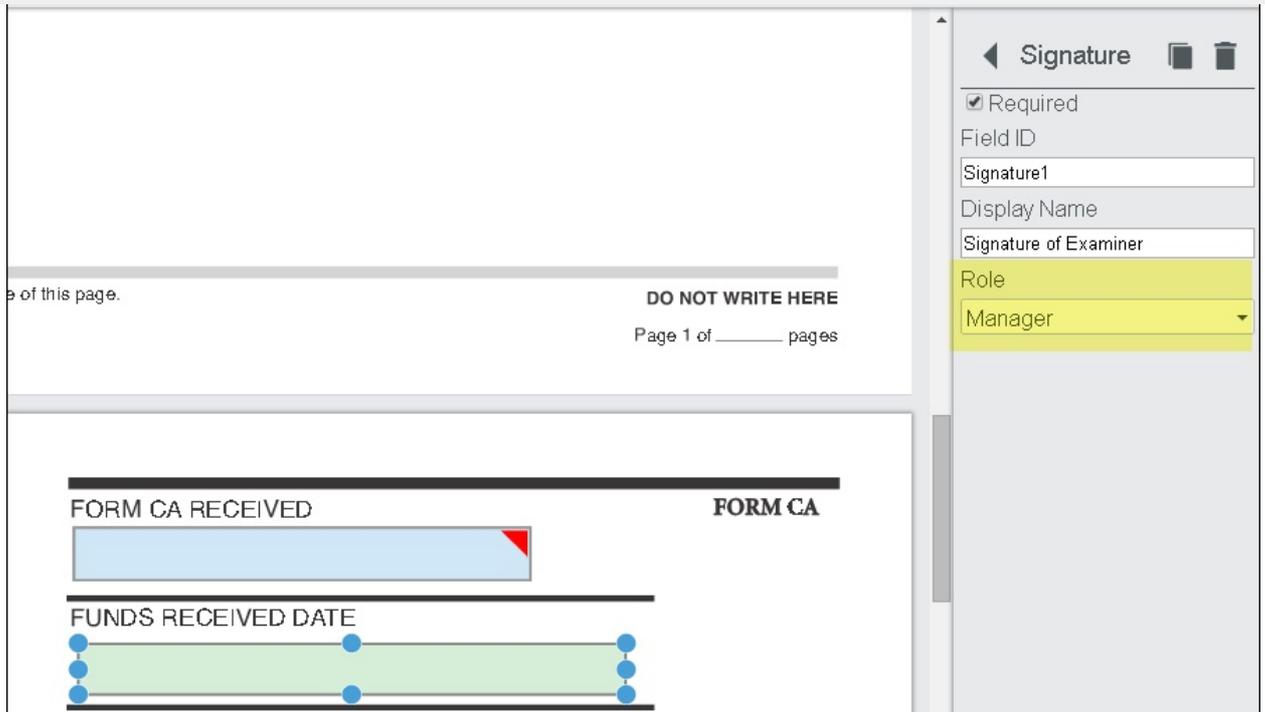
Role  
Employee

Vertical Alignment  
[Icons]

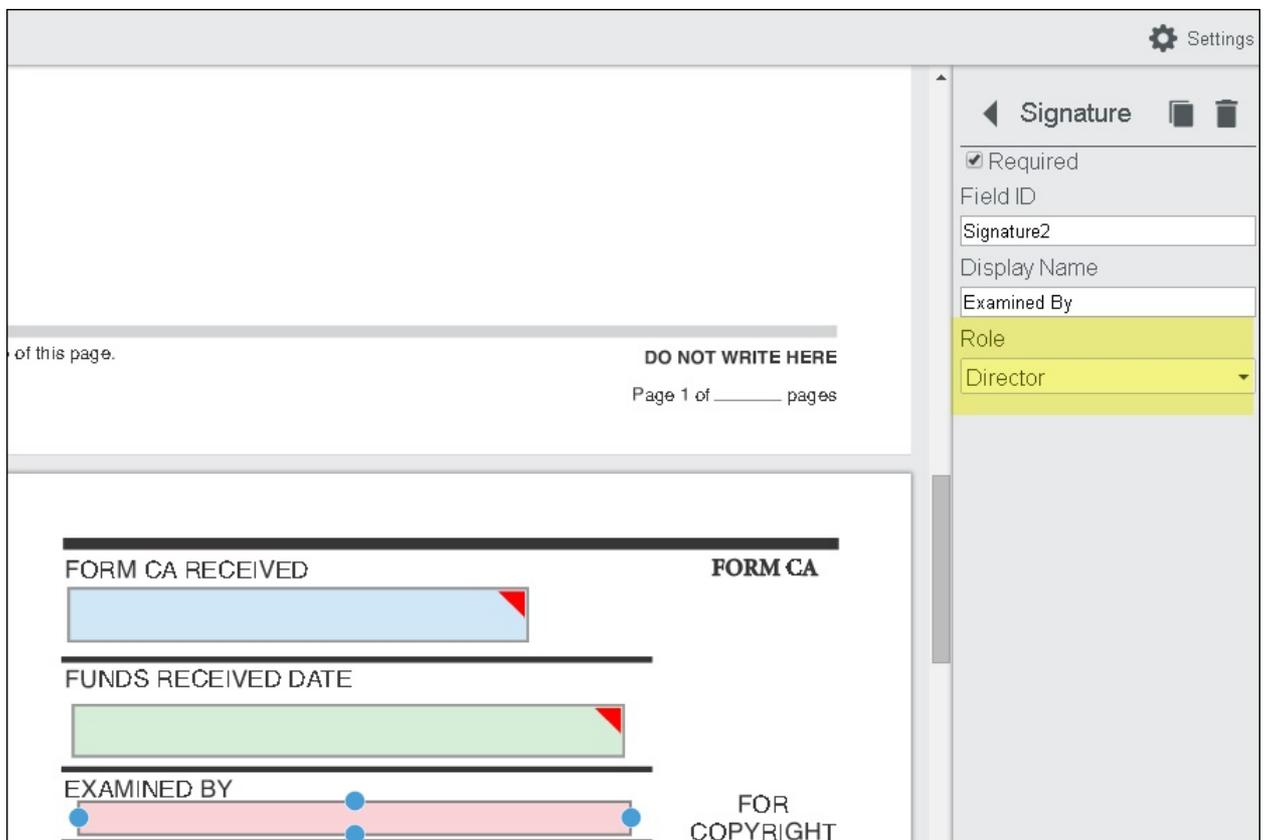
Horizontal Alignment  
[Icons]

Match Size  
[Icons]

2. Click on the **Role** drop-down and select **Employee**. The field turns blue indicating the field is required for an Employee to review and fill out.
3. Click on the **Funds Received Date** field and the Properties Panel is displayed:



4. Click on the **Role** drop-down and select **Manager**. The field turns green indicating the field is required for a Manager to review and fill out.
5. Click on the **Examined By** field and the Properties Panel is displayed:



6. Click on the **Role** drop-down and select **Director**. The field turns red indicating the field is required for a Director

7. Once you have assigned all of the roles to the desired fields, click **Save** at the top of the template.

## Assigning an Input Mask to a Text Box

By adding an Input Mask to a text box, you can require a signer to fill in the text box with specific information, such as a Phone Number, Social Security Number, Date, Zip Code, or Model Number. Valid entries for the Input Mask are: # - Numeric, A - Alphabetic, and \* - Alphanumeric.

Use the following steps to add an Input Mask to a field which requires a telephone number:

1. Create a **text box**.
2. In the Properties Panel, enter **(###)###-####** in the Input Mask field.



3. Click **Save** at the top of the template.

## Use the Form Field Detector

The E-Signature Module's Form Field Detector supports the ability for you to upload a PDF document or raster image with form fields, auto-detect and convert the active fields for the E-Signature Template. You can modify the fields as needed and share the document with another user who will **complete and sign the document**. The Form Field Detector facilitates a fast, convenient and more secure process for working with PDF documents and raster images with form fields.

### Supported Fields

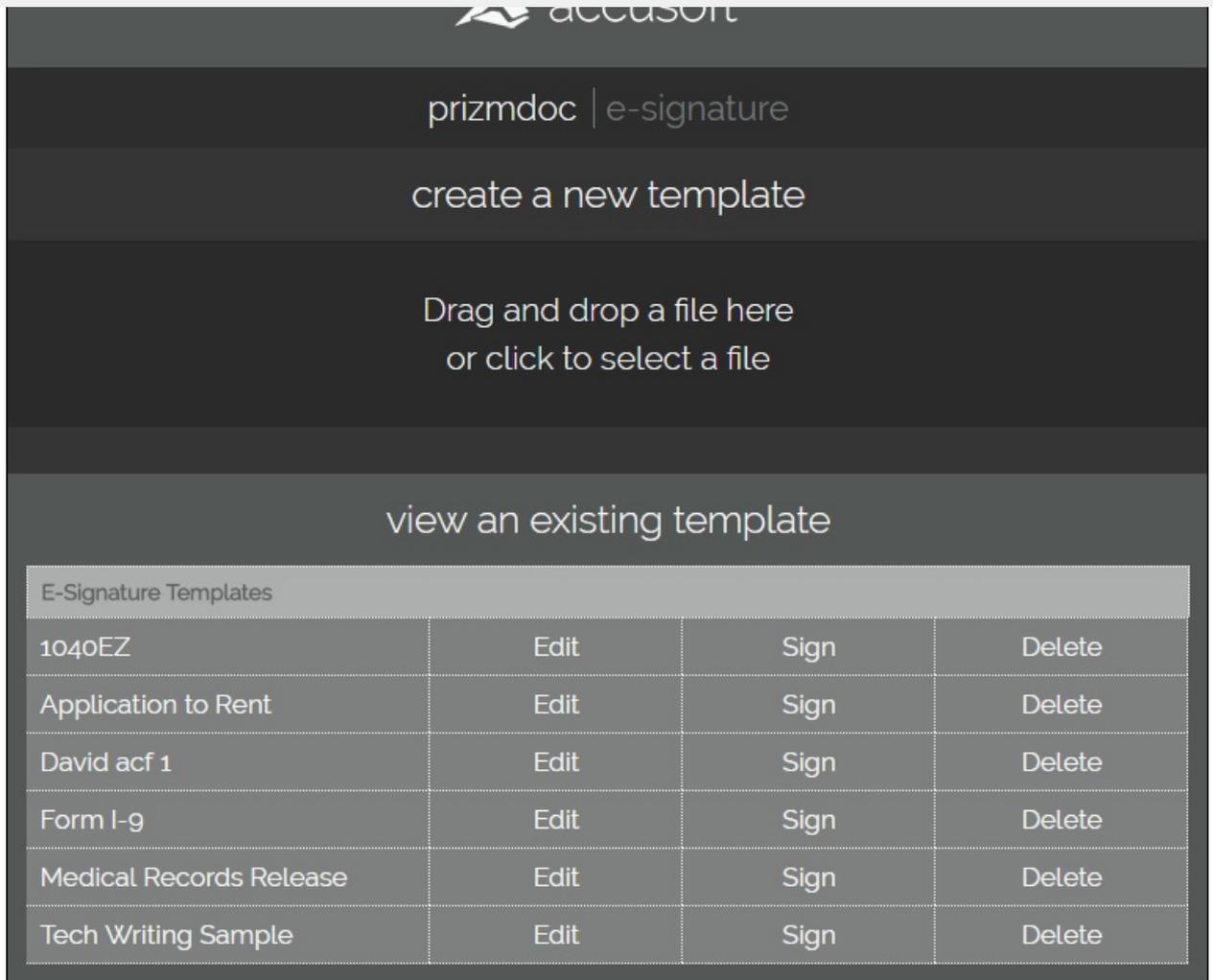
Field Type	Description
Button Fields	Check box and Radio button field types are supported. The Push-button field type is not supported.
Text Fields	Are supported, including limited support for font and color attributes.
Choice Fields	Not supported.
Signature Fields	Are supported.

### Adding a PDF file or Raster Image to the E-Signature Splash Screen

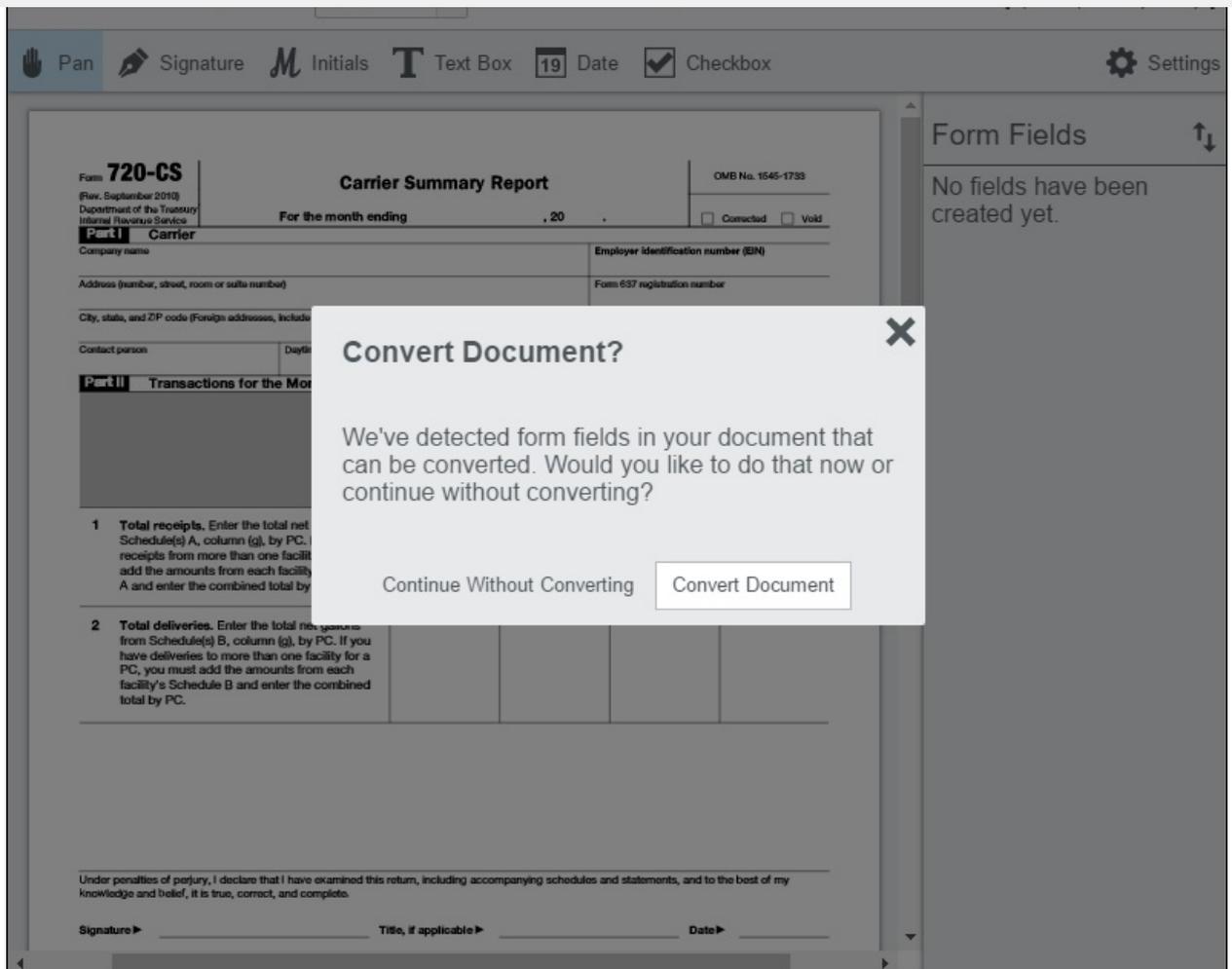
Use the following steps to upload and convert a document using the Form Field Detector:

 The following example uses a PDF file, but the same process can be used for a raster image with form fields.

1. From the splash screen, under **Create a New Template**, drag and drop the **PDF file** that you want to use:

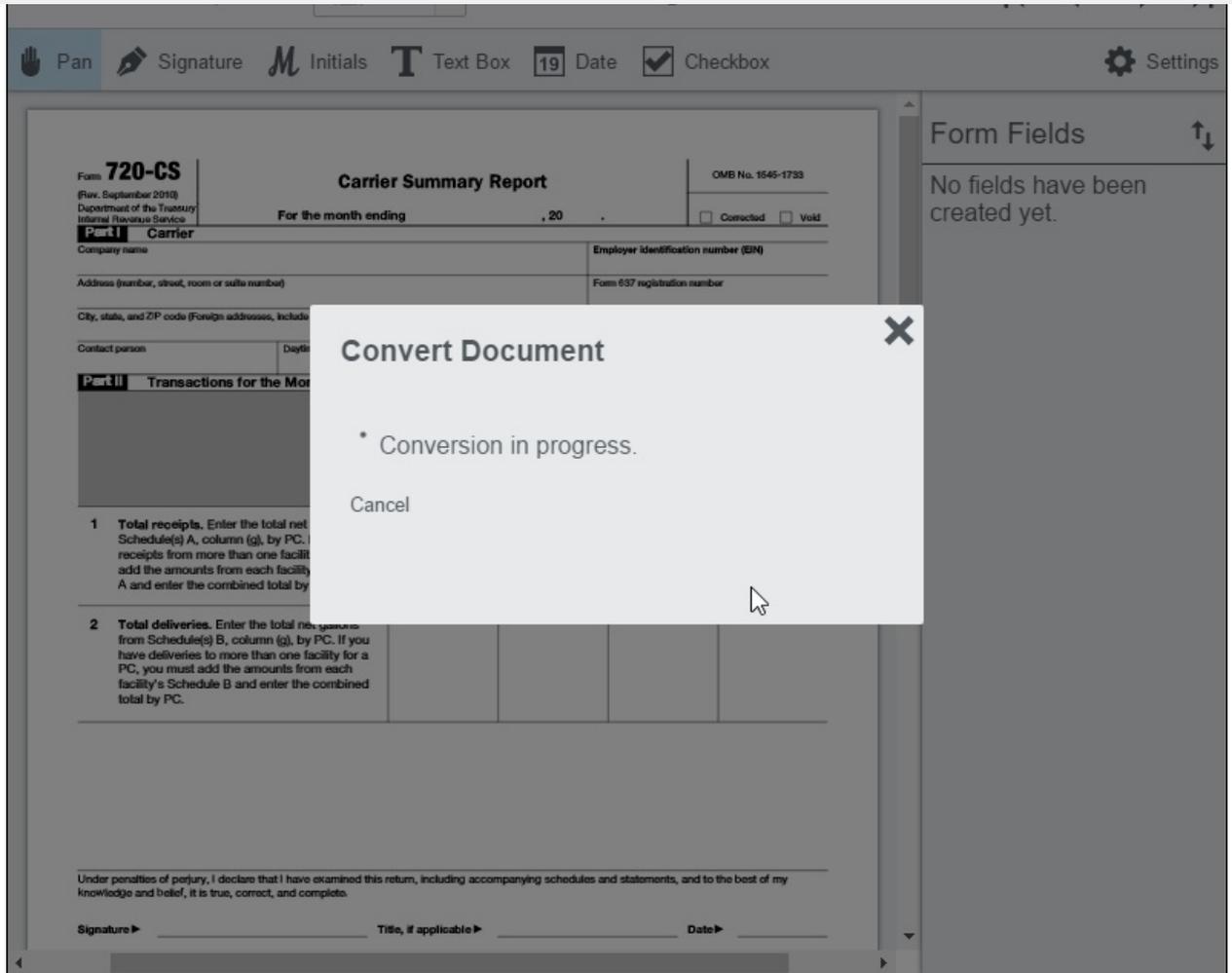


2. The Convert Document dialog displays:

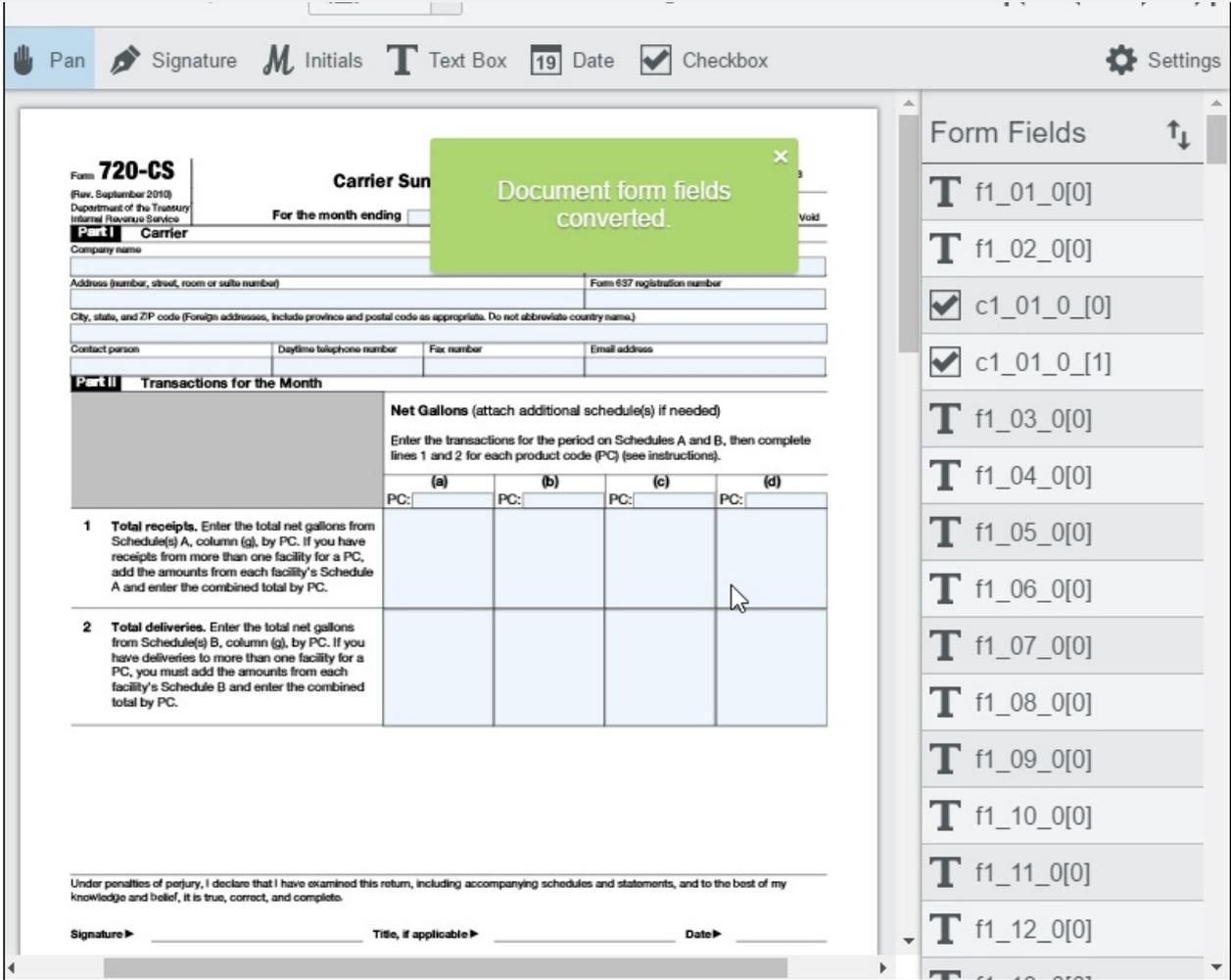


Choose to either **Continue Without Converting** or **Convert Document**. (If you choose Continue Without Converting, the PDF file will open, but the form fields will not be converted.)

3. After you click **Convert Document**, the Convert Document Progress dialog displays:



4. Once the PDF file has successfully converted, the Document Form Fields Converted dialog displays:



5. The PDF file is displayed within the Template Designer and you can click on the converted fields and make changes as desired:

The screenshot displays the PrizmDoc interface with a 'Carrier Summary Report' form. The form includes sections for 'Part I Carrier' (Company name, Address, Contact person) and 'Part II Transactions for the Month' (Net Gallons table). A right-hand sidebar shows the properties for a selected 'Text Box' field, including 'Field ID', 'Display Name', 'Font', 'Color', and 'Character Limit'.

**Form Fields:**

- Form: 720-CS (Rev. September 2010)
- Department of the Treasury Internal Revenue Service
- Carrier Summary Report
- OMB No. 1545-1738
- For the month ending [ ] , 20 [ ] .
- Part I Carrier
- Company name
- Employer identification number (EIN)
- Address (number, street, room or suite number)
- Form 637 registration number
- City, state, and ZIP code (Foreign addresses, include province and postal code as appropriate. Do not abbreviate country name.)
- Contact person
- Daytime telephone number
- Fax number
- Email address
- Part II Transactions for the Month
- Net Gallons (attach additional schedule(s) if needed)
- Enter the transactions for the period on Schedules A and B, then complete lines 1 and 2 for each product code (PC) (see instructions).
- Table with columns (a), (b), (c), (d) and rows for Total receipts and Total deliveries.
- Signature, Title, if applicable, Date

**Field Properties (Text Box):**

- Field ID: Text3
- Display Name: f1\_03\_0[0]
- Font: Use Global Setting
- Color: [Blue]
- Character Limit: 0

6. After you have finished making your changes, you can name the template and save it.

Refer to the following topic for possible error messages:

- [Form Field Detector Error Messages](#)

## Making Additional Changes to Your Template

For details on how to make additional changes to your template, refer to the following topics:

- [How to Set up a Basic Template](#)
- [How to Add Features to a Template](#)

## Signing the Template

For information on how to sign a PDF that has been converted to an E-Signature Template, refer to the following topic:

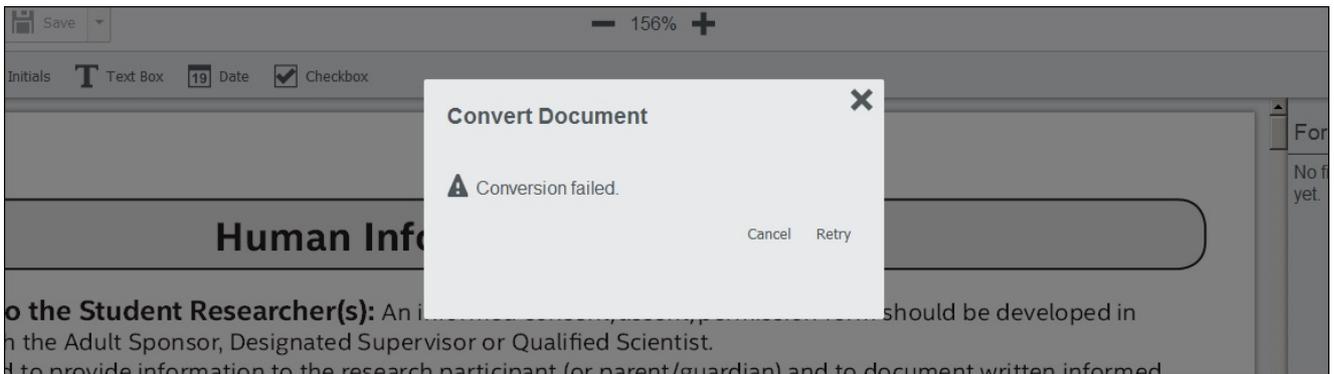
- [Sign a Form Converted by the Form Field Detector](#)

You may encounter error messages for the following reasons:

- Generic error detecting form fields in the uploaded document.
- Form Field Detector feature is not licensed.
- The Form Field Detector fields do not have a matching PrizmDoc E-Signature field.
- Documents that contain XFA fields are not currently supported.

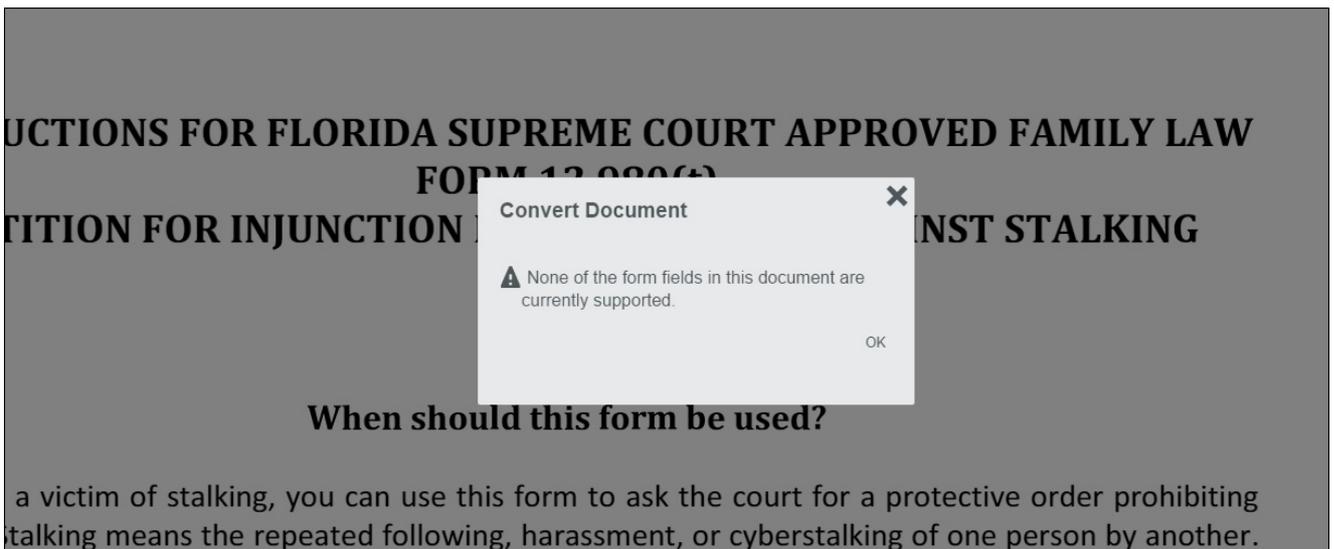
### Example Error Messages

#### Conversion failed:



If you are unsure whether you have the Form Field Detector feature licensed, refer to the [Form Field Detector troubleshooting page](#) to determine if the error is related to licensing or not.

#### Form fields are not supported:



#### XFA fields are not currently supported:



ox 19 Date  Checkbox

INT THE FOLLOWING INFORMATION LEGIBLY AS THIS DOCUMENT IS  
ORDER TO PREPARE TH NOTE: NO  
MAY BE MADE TO THIS DER AFTER THE  
(AS BEEN ACCEPTED.)

NUMBER: \_\_\_\_\_ DATE OF SALE: \_\_\_\_\_

NAME IN WHICH THE CERTIFICATE OF TITLE IS TO BE ISSUED:

This document contains XFA fields. Documents that contain XFA fields are currently not supported.

## Signers - Fill out a Form

This section contains information on how to fill out forms:

- [Tools Reference](#)
- [Fill Out a Form](#)
- [Sign a Form with Multiple Roles](#)
- [Sign a Form Converted by the Form Field Detector](#)

## Tools Reference

This section covers the following features:

- [Example of a Form in the E-signature Viewer](#)
- [Form Name](#)
- [Checklist](#)
- [Zoom In / Zoom Out](#)
- [Page Navigation](#)
- [Example of a Required Field](#)
- [Example of an Optional Field](#)
- [Progress Bar](#)
- [Download Signed Form](#)

### Example of a Form in the E-signature Viewer

The following example shows a form displayed in the main e-signature viewer:

Copyright Office fees are subject to change. For current fees, check the Copyright Office website at [www.copyright.gov](http://www.copyright.gov), write the Copyright Office, or call (202) 707-3000 or 1-877-478-0778.

Privacy Act Notice: Sections 408-410 of title 17 of the United States Code authorize the Copyright Office to collect the personally identifying information requested on this form in order to process the application for copyright registration. By providing this information you are agreeing in routine uses of the information that include publication to give legal notice of your copyright claim as required by 17 U.S.C. 5705. It will appear in the Office's online catalog. If you do not provide the information requested, registration may be refused or delayed, and you may not be entitled to certain relief, remedies, and benefits under the copyright law.

**Form CA**  
For Supplementary Registration  
UNITED STATES COPYRIGHT OFFICE  
REGISTRATION NUMBER

TX | TRU | PA | PAU | VA | VAI | SR | SRU | RE

EFFECTIVE DATE OF SUPPLEMENTARY REGISTRATION

Month Day Year

DO NOT WRITE ABOVE THIS LINE. IF YOU NEED MORE SPACE, USE A SEPARATE CONTINUATION SHEET.

**A**

Title of Work ▼

Registration Number of the Bank Registration ▼

Year of Bank Registration ▼

Progress 2 optional left

Download Signed Form

## Form Name

The following example shows the Form Name:

Tech Writing Sample

- This is not an editable field.

## Checklist

The following example shows the Checklist Icon:

Checklist

1. When you click on the **Checklist** icon, the Checklist expands:

<input checked="" type="checkbox"/>	Signature of Examiner
<input checked="" type="checkbox"/>	Initials of Reviewer
<input checked="" type="checkbox"/>	Important Text
<input checked="" type="checkbox"/>	Date Received
<input type="checkbox"/>	Registration Added
<input type="checkbox"/>	Registration not added

- The **red star** indicates a **required** field.
  - The **grey circle** indicates an **optional** field.
2. When you fill out a required field, the red star turns to a green check mark. When you fill out an optional field, the grey circle also becomes a green check mark:

<input checked="" type="checkbox"/>	Signature of Examiner
<input checked="" type="checkbox"/>	Initials of Reviewer
<input checked="" type="checkbox"/>	Important Text
<input checked="" type="checkbox"/>	Date Received
<input checked="" type="checkbox"/>	Registration Added
<input type="checkbox"/>	Registration not added

### Zoom In / Zoom Out

The following example shows the Zoom In / Zoom Out Toolbar:



- Minus Sign - click to zoom out
- Percentage - displays the viewing percentage
- Plus Sign - click to zoom in

### Page Navigation

The following example shows the Page Navigation Toolbar:



- First Page - click to go to the first page of the form
- Previous Page - click to go to the previous page
- Next Page - click to go to the next page
- Last Page - click to go to the last page of the form

### Example of a Required Field

The following example shows a field with a red star which indicates that it must be filled out:

<b>Title of Work ▼</b> <input type="text"/>
--

 Required Checkboxes will not have a red star next to them on the form, but they will display in the Checklist as required.

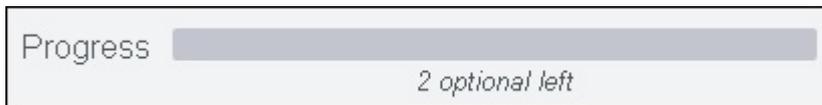
### Example of an Optional Field

<b>REFERENCE TO THIS REGISTRATION ADDED TO BASIC REGISTRATION</b> <input type="checkbox"/> YES <input type="checkbox"/> NO
--

## Progress Bar

The following examples show the different stages of the progress bar:

- The Progress Bar is grey prior to you filling out the form fields:



- When you start filling out some of the fields, the Progress Bar turns blue:



- After you have filled out all of the required fields, the Progress Bar turns green and you can download the signed form:

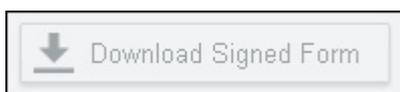


 If you exit a form without filling in all the fields, a message appears, "Confirm Navigation - Warning: You have made changes, download before continuing. Are you sure you want to leave this page?" Click 'Leave this Page' to exit without downloading changes or click 'Stay on this Page' to fill out the remaining fields and download your changes.

## Download Signed Form

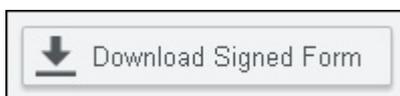
The following examples show the Download Signed form button:

- The Download Signed Form button is grey prior to you filling out the template fields:



 The Download Signed Form button remains inactive until you fill out all the required fields.

- When you fill out all the required fields, the Download Signed Form button is activated:



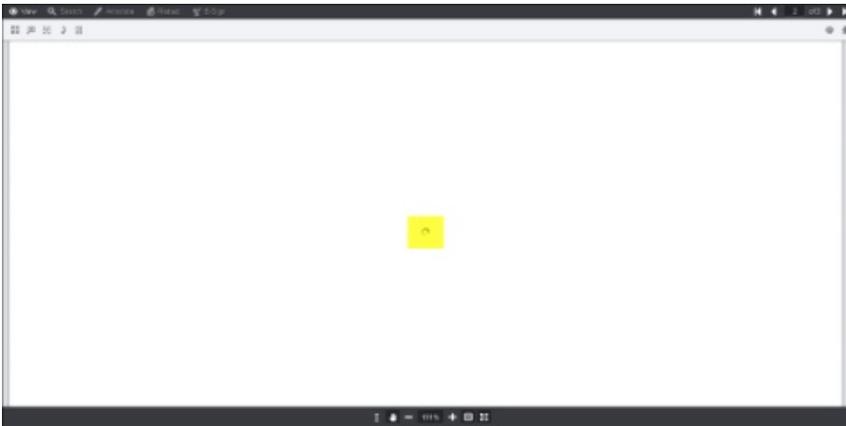
## Fill Out a Form

This topic contains the options and steps for filling out a form:

- Working with Large Files
- Opening a Form
- Working with Signatures
- Working with Initials
- Working with Text Boxes
- Working with Dates
- Working with Checkboxes

### Working with Large Files

Note that if you open, scroll-through or use set page number for a very large file, you may see the spinning circle (highlighted in yellow below) displayed in the middle of the Viewer while the document loads:



### Opening a Form

1. From the splash screen, under View an Existing Template section, click on **View** next to the template you want to fill



out:

2. The template is displayed in the Viewer and the Checklist pane displays:

3. The required fields have a red star next to them and the optional fields have a grey circle.

 Required Checkboxes will not have a red star next to them on the form, but they will display in the Checklist as required.

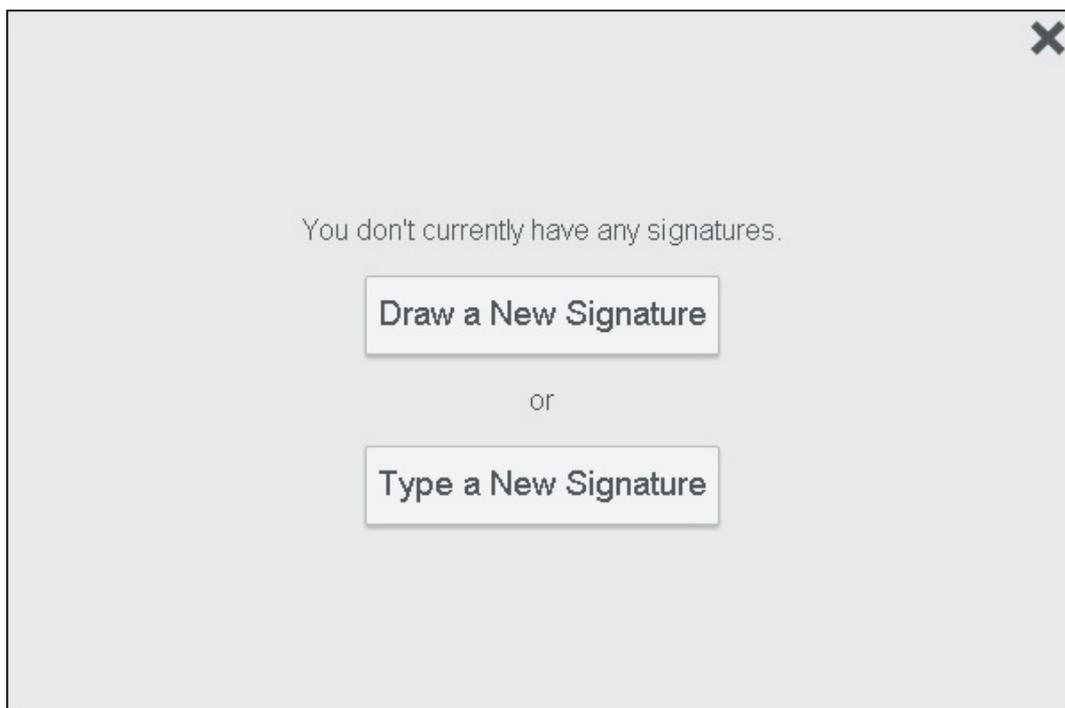
4. Starting with the first item in the Checklist, you can press the **Tab** key to move through the fields on the form. You will need to use the mouse to choose a date, draw a signature, draw initials or select a checkbox.

## Working with Signatures

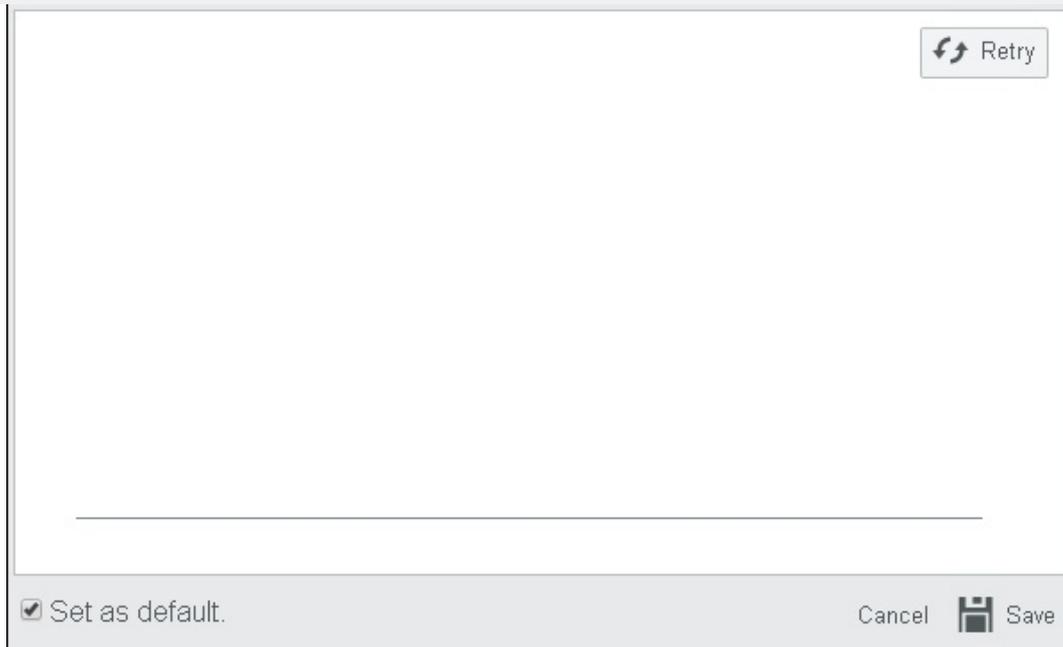
You can create both drawn and typed signatures to use throughout the document. You can also remove them from the document or delete them permanently and start over with new signatures. This section covers these options in detail.

### Drawing a New Signature

1. Click on the **first item** in the list. In this example, click on **Signature of Examiner**. The form focus moves to the Signature field.
2. Click in the **Signature** field and the Draw or Type a New Signature dialog box displays:



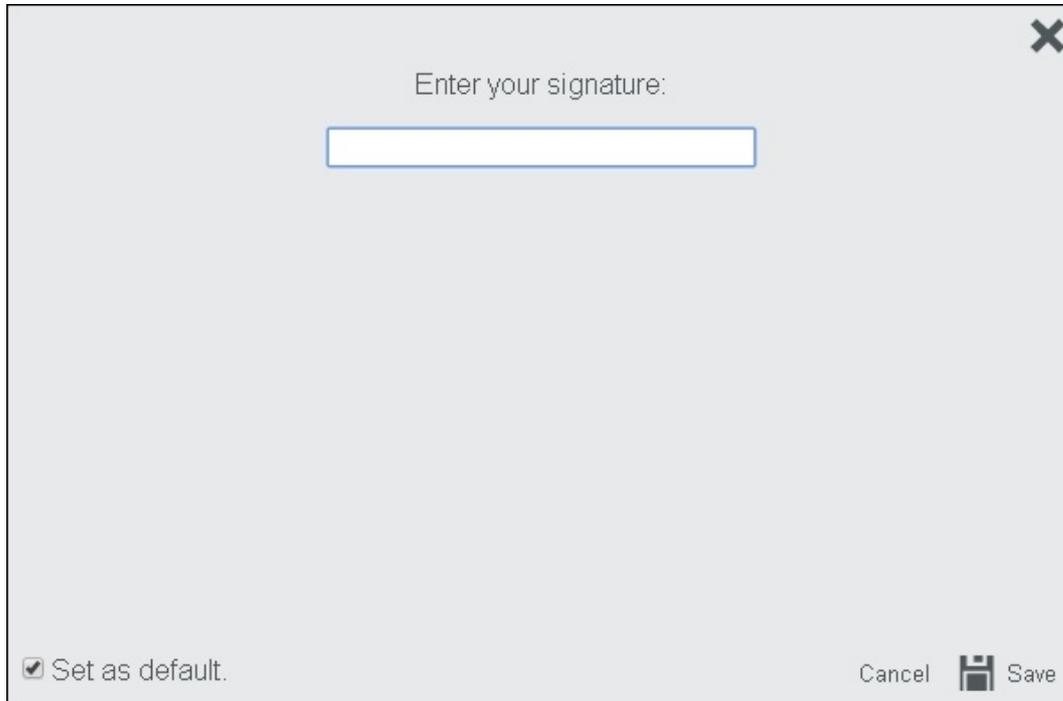
3. To draw a new signature, click on **Draw a New Signature**. The Draw Signature dialog box displays:



4. With your mouse, **draw your signature** above the line. You have the option to Retry, Set as default, Cancel or Save. When you are finished, click **Save**. Your signature is placed in the Signature field.

### Typing a New Signature

1. To type a new signature, click on **Type a New Signature**. The Type a New Signature dialog box displays:



2. Type your **signature** in the Text box. Your signature displays in a variety of fonts:



3. You have the option to Set as default, Cancel or Save. Click on the signature you want and click **Save**. Your signature is placed in the Signature field.

### Applying a Signature

1. To apply a different signature, click in the **Signature** field. The Signature dialog box displays:



2. Scroll to the signature you want to use and click **Apply to document**.
3. Click **Done**.

### Removing a Signature

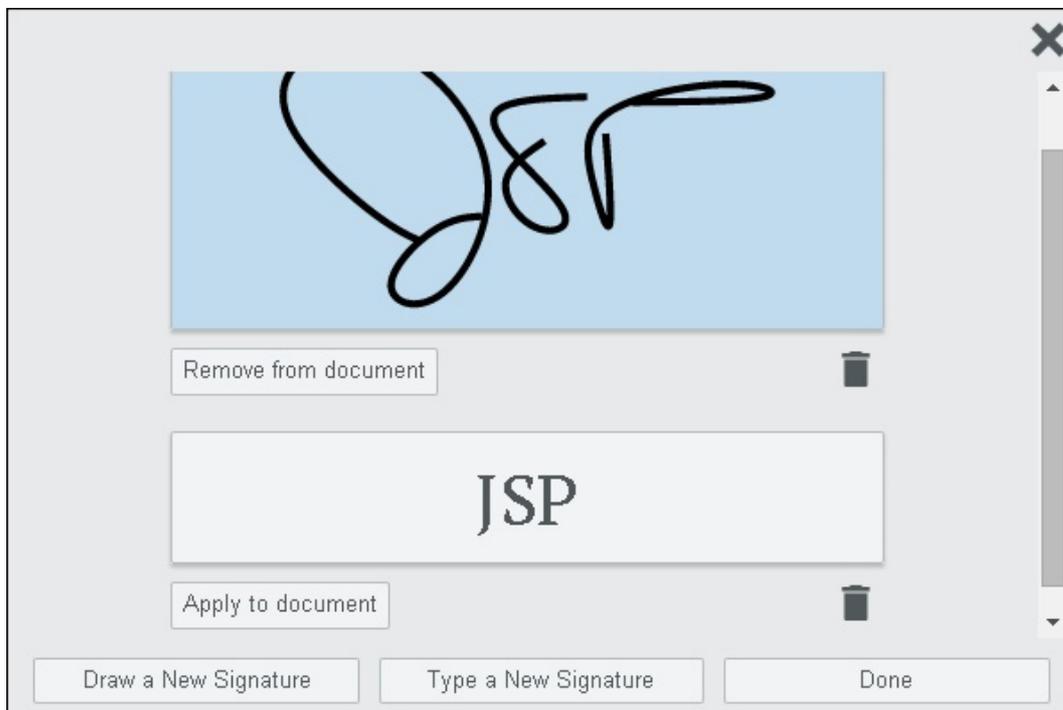
1. To remove a signature from the document, click in the **Signature** field. The Signature dialog box displays:



2. Scroll to the signature you want to remove and click on **Remove from document** (this removes the signature from the document, but keeps it available to use again if you want).
3. Click **Done**.

### Deleting a Signature

1. To delete a signature from the document, click on the **Signature** field. The Signature dialog box displays:



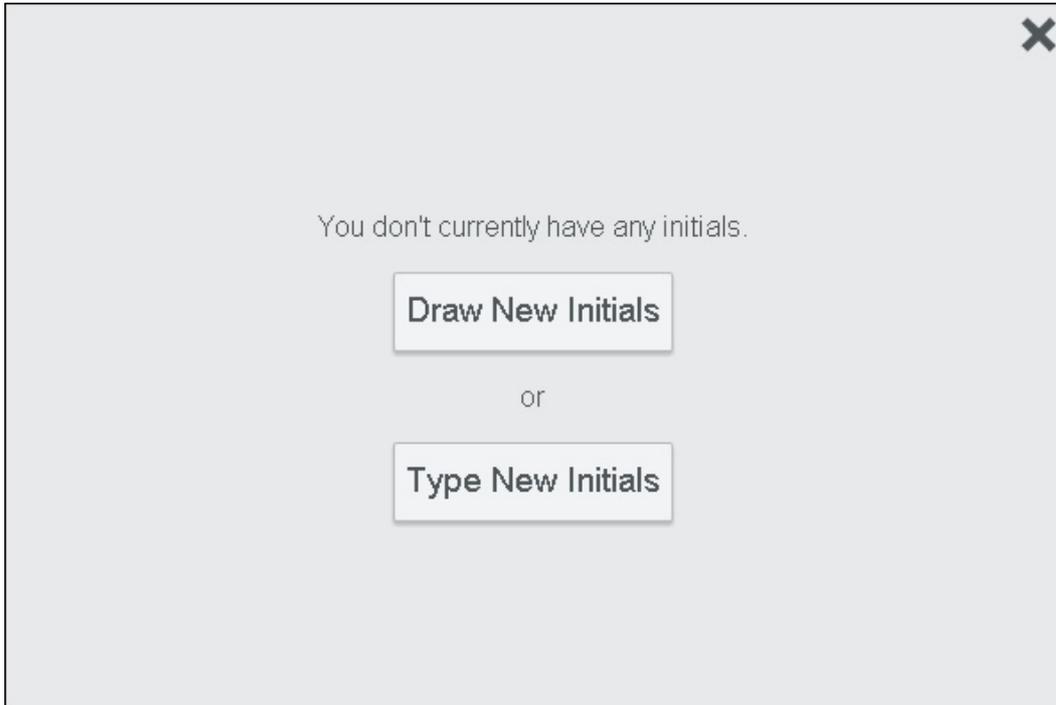
2. Click on the **Delete** (trash can) icon.
3. The signature is deleted from the e-signature form completely and is not available for use again.
4. Click **Done**.

### Working with Initials

or delete them permanently and start over with new initials. This section covers these options in detail.

### Drawing New Initials

1. Click on the **second item** in the list. In this example, click on **Initials of Reviewer**. The form focus moves to the Initials field.
2. Click in the Initials field and the Draw or Type New Initials dialog box displays:



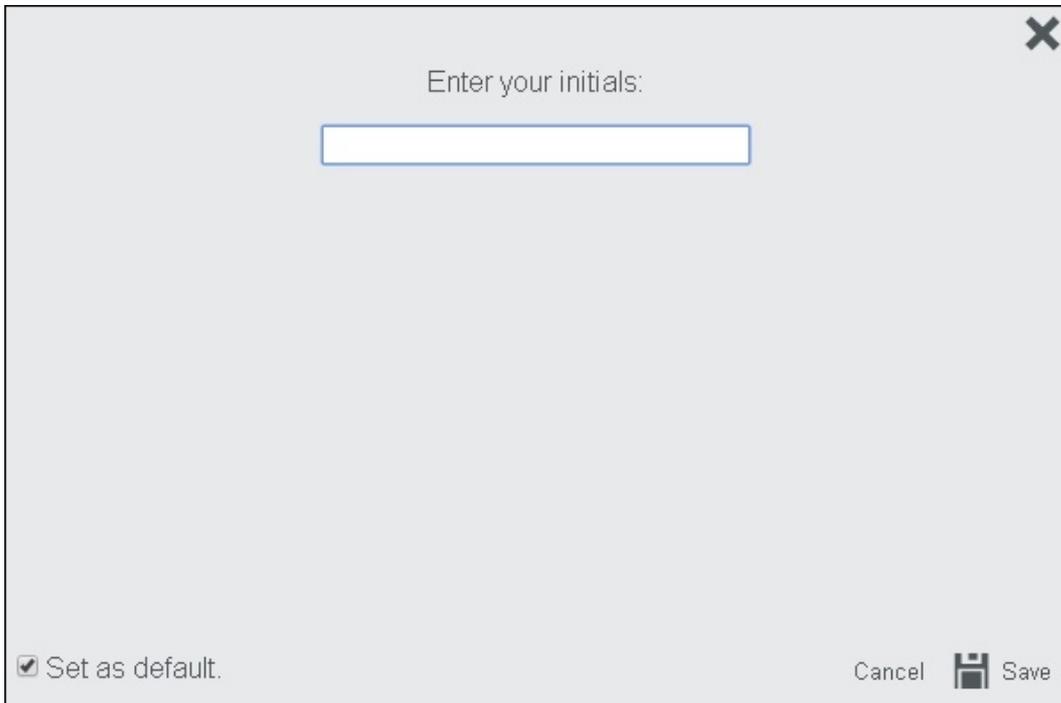
3. To draw new initials, click on **Draw New Initials**. The Draw Initials dialog box displays:



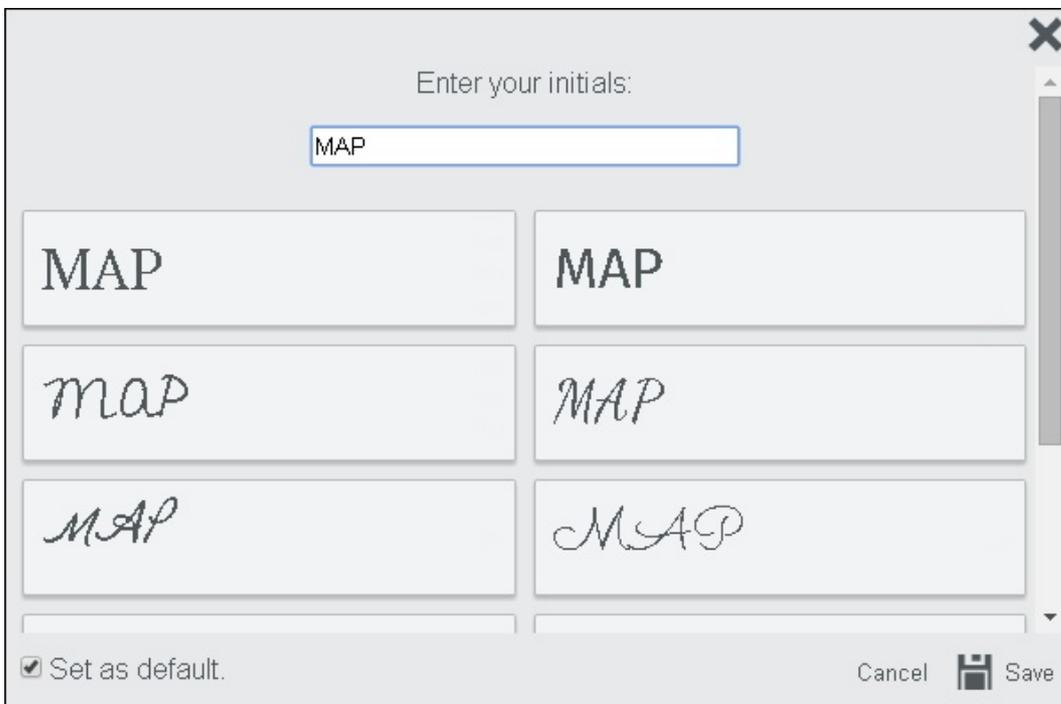
4. With your mouse, **draw your initials** above the line. You have the option to Retry, Set as default, Cancel or Save.

## Typing new initials

1. To type new initials, click on **Type New Initials**. The Type a New Initials dialog box displays:



2. Type your **initials** in the Text box. Your initials display in a variety of fonts:



3. You have the option to Set as default, Cancel or Save.
4. Click on the initials you want and click **Save**. Your initials are placed in the Initials field.

## Applying Initials

1. To apply different initials, click in the **Initials** field. The Initials dialog box displays:



3. Scroll to the initials you want to use and click **Apply to document**.
4. Click **Done**.

### Removing Initials

1. To remove initials from the document, click in the **Initials** field. The Initials dialog box displays:



2. Scroll to the initials you want to remove and click on **Remove from document** (this removes the initials from the document, but keeps them available to use again if you want).
3. Click **Done**.

### Deleting Initials

1. To delete initials from the document, click on the Initials field. The Initials dialog box displays:

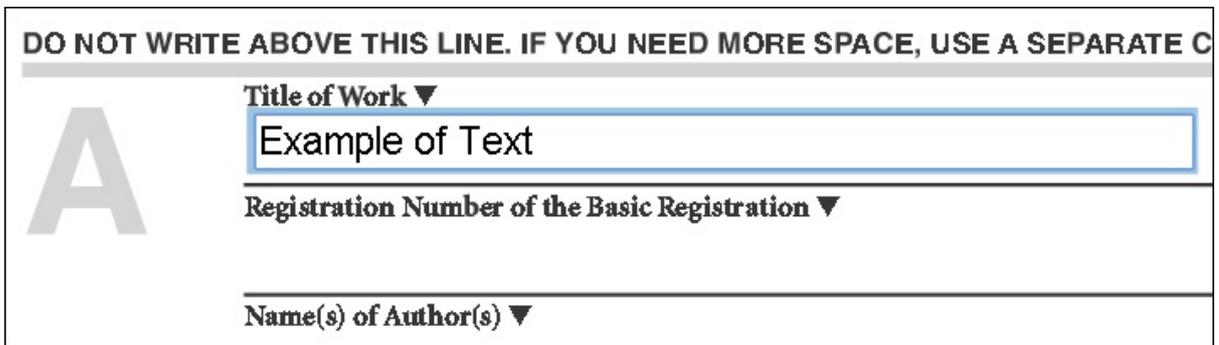


2. Click on the **Delete** (trash can) icon.
3. The initials are deleted from the e-signature form completely and are not available for use again.
4. Click **Done**.

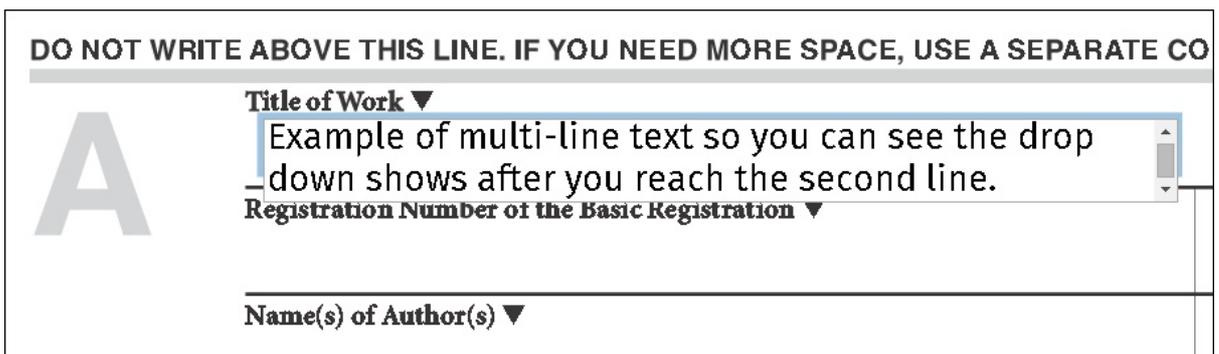
## Working with Text Boxes

### Adding Text

1. To add text, click in the **Text Box** field:



2. Type in the required or optional text. Note that if the text box allows multiple lines of text, you will see a scroll-bar appear on the right side of the text box as you are typing:



the field showing the expected format of the text and you will only be able to enter certain characters. In this example, you will only be able to enter numbers:

<b>Occupation</b>	<b>Daytime phone number</b> ( ) -
<b>PTIN's occupation</b>	If the IRS sent you an Identity Protection PIN, enter it here (see inst.)
<b>Date</b>	<input type="checkbox"/> <b>Check if self-employed</b> <input type="checkbox"/> <b>PTIN</b>

An Input Mask field could be for any of the following: Date, Social Security Number, Phone Number, Zip Code, or Model Number.

- Click outside of the text box and go to the next field.

**⚠** If the box outline turns red, you have exceeded the character limit of the text box:

**DO NOT WRITE ABOVE THIS LINE. IF YOU NEED MORE SPACE, USE A SEPARATE C**

**Title of Work ▼**  
Example of Text Too Long ★

**Registration Number of the Basic Registration ▼**

**Name(s) of Author(s) ▼**

You must edit/delete some of the text within the text box.

## Working with Dates

### Adding Dates

- To add a date, click in the **Date** field. The date picker displays:

his

◀ F      **May 2015**      ▶ M

Sun	Mon	Tue	Wed	Thu	Fri	Sat
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

**FUNDS RECEIVED DATE**

- Click on the **date** you want and the date is entered into the Date field.

### Selecting Checkboxes

Depending on how the form was designed, you will be able to select either:

- One checkbox out of group
- Several checkboxes in a group

1. To select a checkbox, click in the **Checkbox**. The check mark displays in the checkbox:



REFERENCE TO THIS REGISTRATION ADDED TO  
BASIC REGISTRATION  YES  NO

 Required Checkboxes will not have a red star next to them on the form, but they will display in the Checklist as required.

## Sign a Form with Multiple Roles

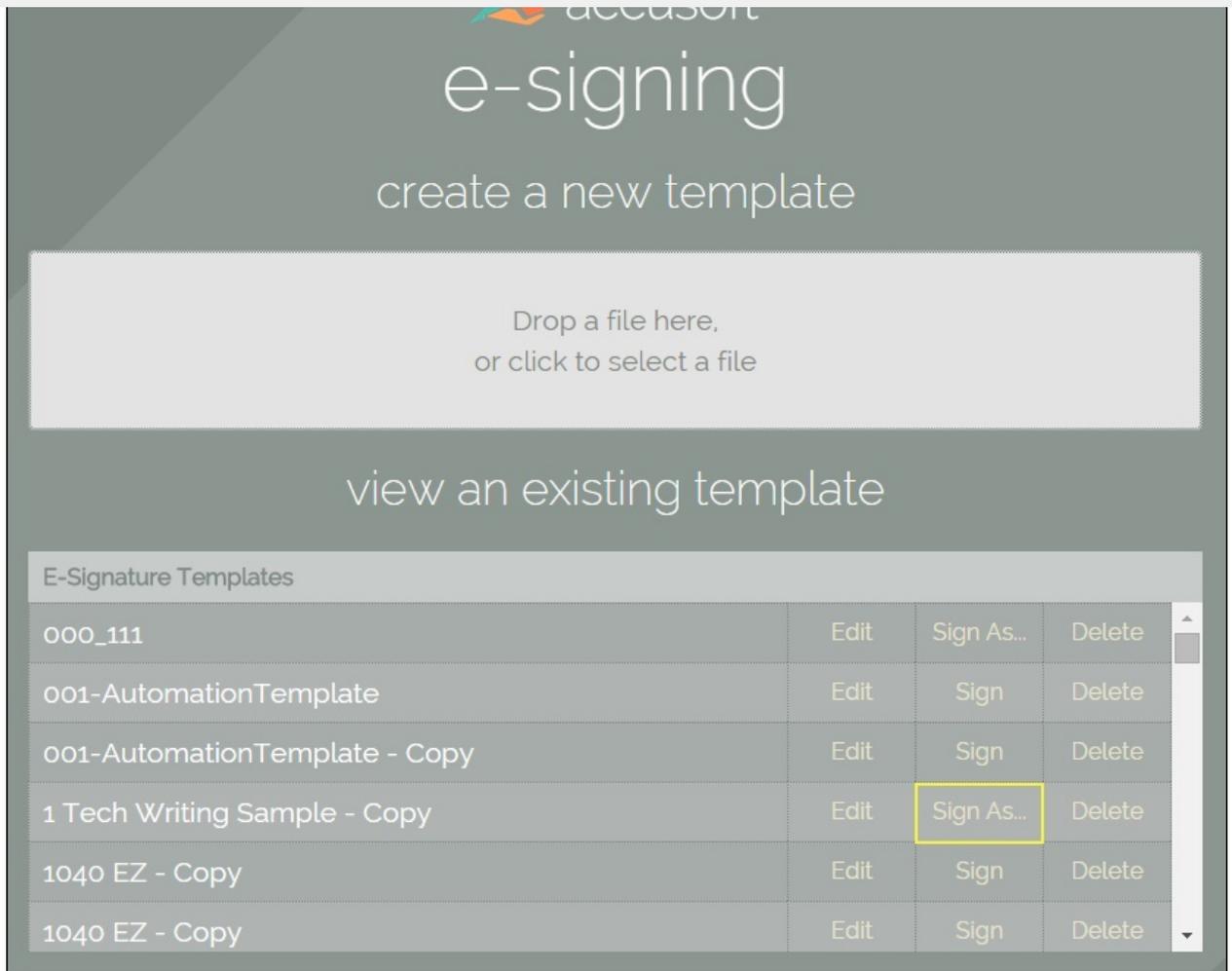
This topic contains the steps for signing a form that has multiple roles. For example, an Employee, a Manager and a Director all need to review and sign a form. Note that for ease-of-use, after each role/reviewer has filled out the required fields and the Download button is enabled, the user gets a copy of the form and another copy is stored so that the remaining roles/reviewers can see the information already added to the form.

To review and fill out a form with multiple roles:

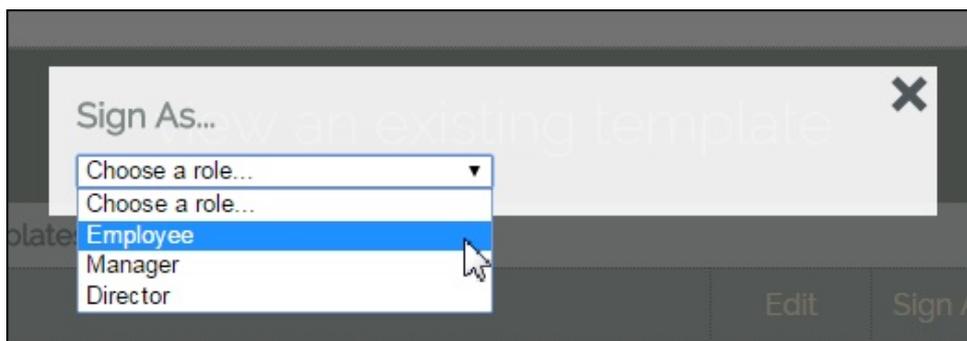
### Employee Example

In this example, the Employee role/reviewer is going to open and fill out the form first.

1. From the **E-signing** Splash Screen, under **View an Existing Template**, click **Sign As...** next to the form you want to fill out:



2. The Sign As... dialog is displayed. For this example, choose the **Employee** role:



3. The form is displayed in the Viewer and all the fields that are required for the **Employee** are highlighted in **blue**:

09/17/2015

---

FUNDS RECEIVED DATE

---

EXAMINED BY

---

CORRESPONDENCE

---

REFERENCE TO THIS REGISTRATION ADDED TO  
BASIC REGISTRATION  YES  NO

---

4. Once all the required fields are filled out, the **Download Signed Form** button is enabled:

Progress

 Download Signed Form

5. Click the **Download Signed Form** button and the burn-in process begins. Once the burn-in is complete, the **Document ready to download** dialog is displayed:

**Download Signed Form** ✕

Document ready to download.

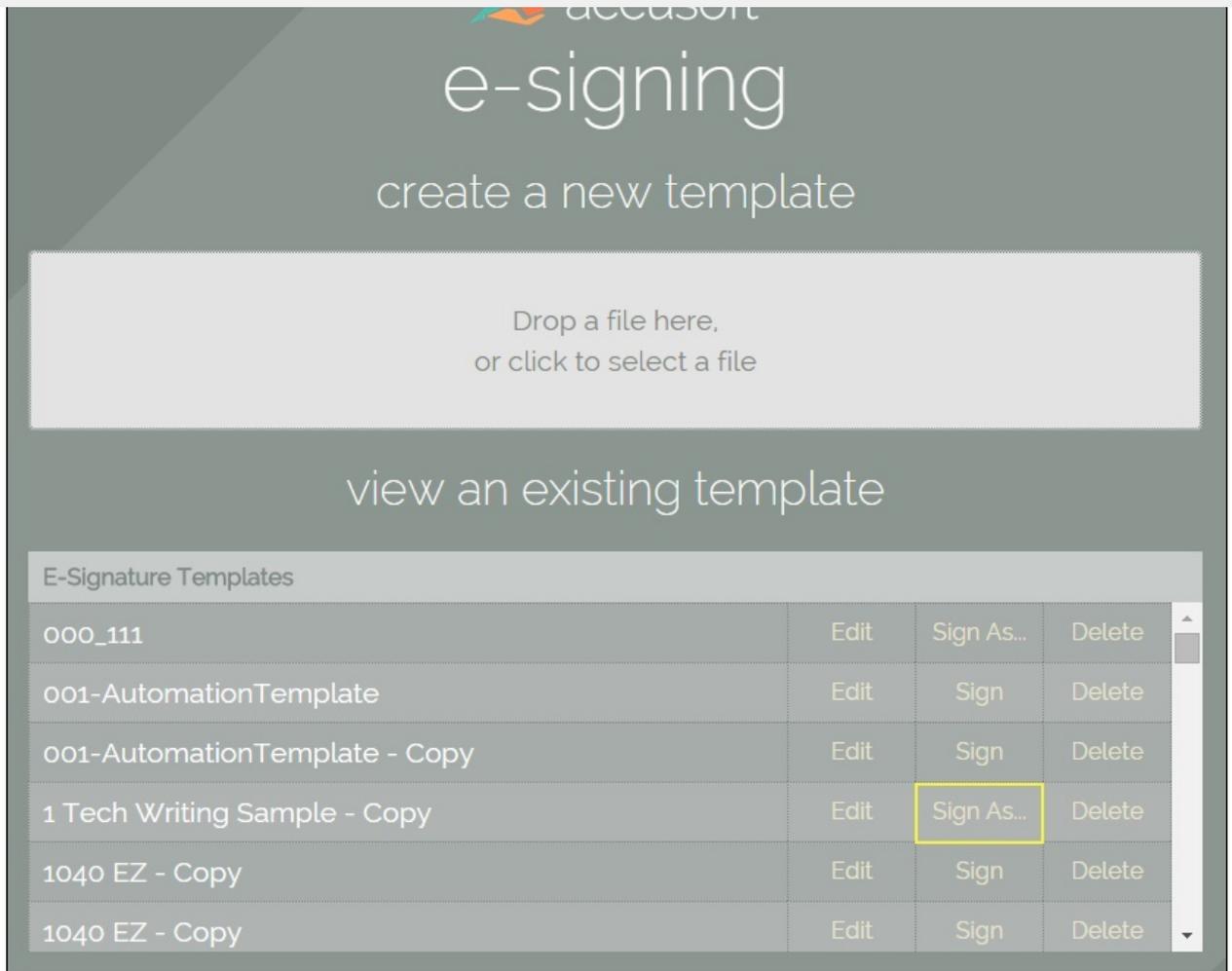
Cancel Save

6. Click **Save** and a copy of the file is available to save to the computer. Note that the Employee's changes have also been saved in a copy of the form so the next role/reviewer can open the form and see their edits.

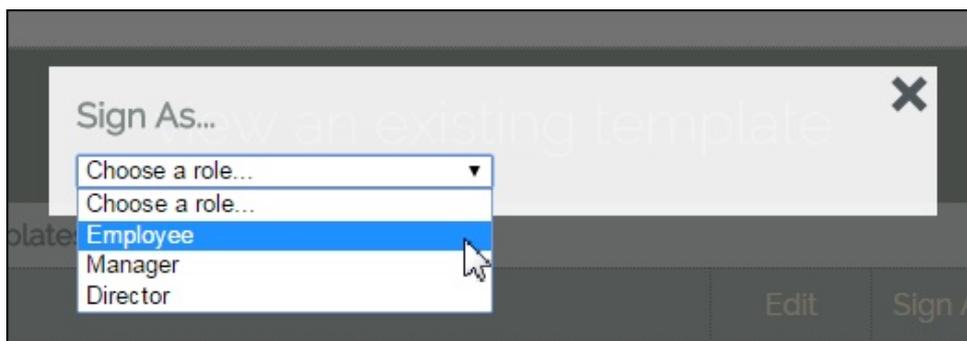
### Manager Example

In this example, the Manager role/reviewer is going to open, review and fill out the form after the Employee (described in the steps above).

1. From the **E-signing** Splash Screen, under **View an Existing Template**, click **Sign As...** next to the form you want to review and fill out:



2. The Sign As... dialog is displayed. For this example, choose the **Manager** role:



3. You can **drop a burned document** in the dialog or click to **select a file**. The form is displayed in the Viewer and all the fields that are required for the **Manager** are highlighted in **green**:

09/17/2015

---

FUNDS RECEIVED DATE

Bob Smith

---

EXAMINED BY

---

CORRESPONDENCE

---

REFERENCE TO THIS REGISTRATION ADDED TO  
BASIC REGISTRATION  YES  NO

- Once all the required fields are filled out, the **Download Signed Form button** is enabled:

Progress

 Download Signed Form

- Click the **Download Signed Form button** and the burn-in process begins. Once the burn-in is complete, the **Document ready to download** dialog is displayed:

**Download Signed Form** ✕

Document ready to download.

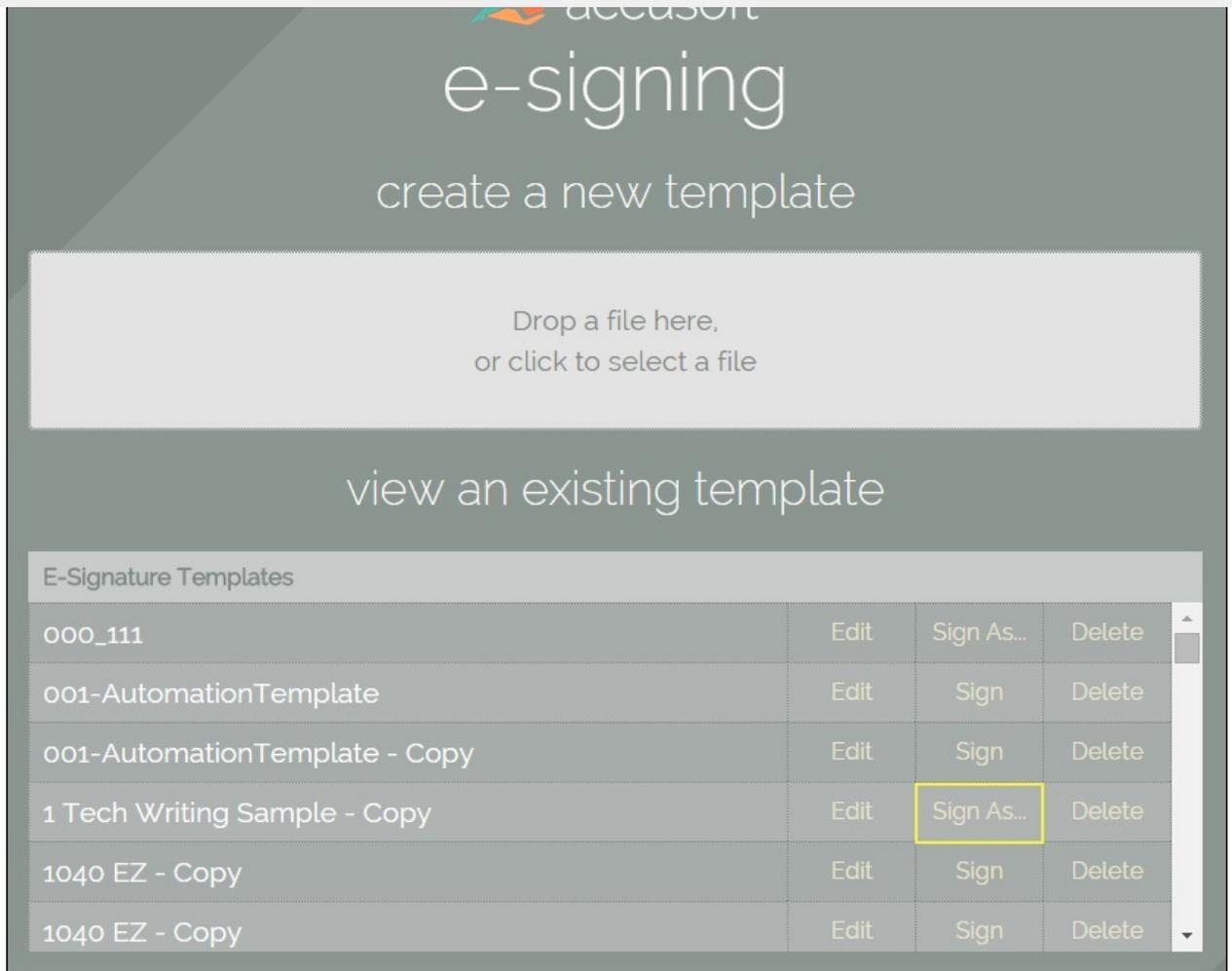
Cancel Save

- Click **Save** and a copy of the file is available to save to the computer. Note that the Manager's changes have also been saved in a copy of the form so the next role/reviewer can open the form and see their edits.

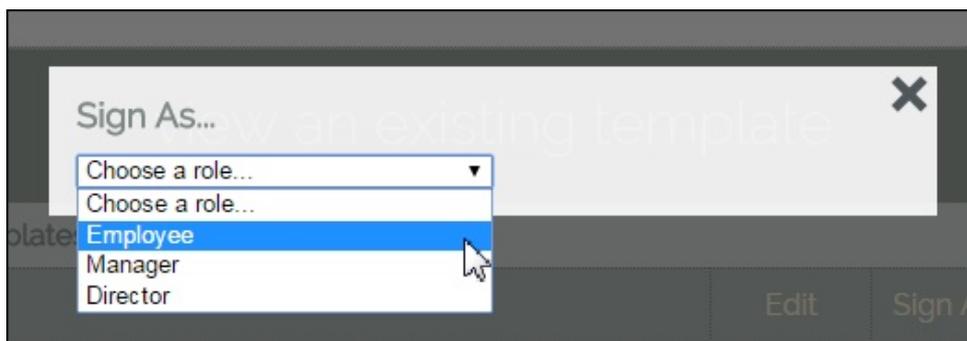
### Director Example

In this example, the Director role/reviewer is going to open, review and fill out the form after the Manager (described in the steps above).

- From the **E-signing** Splash Screen, under **View an Existing Template**, click **Sign As...** next to the form you want to review and fill out:



2. The Sign As... dialog is displayed. For this example, choose the **Director** role:



3. You can **drop a burned document** in the dialog or click to **select a file**. The form is displayed in the Viewer and all the fields that are required for the **Director** are highlighted in **red**:

09/17/2015

FUNDS RECEIVED DATE

Bob Smith

EXAMINED BY Sue Miller

CORRESPONDENCE

REFERENCE TO THIS REGISTRATION ADDED TO BASIC REGISTRATION  YES  NO

4. Once all the required fields are filled out, the **Download Signed Form button** is enabled:

Progress  Download Signed Form

5. Click the **Download Signed Form button** and the burn-in process begins. Once the burn-in is complete, the **Document ready to download** dialog is displayed:

**Download Signed Form** [X]

Document ready to download.

Cancel Save

6. Click **Save** and a copy of the file is available to save to the computer. Note that the Director's changes have also been saved in a copy of the form so the next role/reviewer can open the form and see their edits.

Repeat these steps for each additional role that is required for your document review process.

## Sign a Form Converted by the Form Field Detector

The E-Signature Module's **Form Field Detector** auto-detects forms in a PDF or raster image document and facilitates the conversion to an E-Signature template. You can open the E-Signature template, fill out the form, sign and download it to share with other users. This feature creates a fast, convenient and more secure process for working with PDF documents.

### Viewing an E-Signature Template from the E-Signature Splash Screen

Use the following steps to view, edit, sign and download a document using the Form Field Detector:

The screenshot displays the PrizmDoc e-signature interface. At the top, the Accusoft logo is visible. Below it, the text 'prizmdoc | e-signature' is shown. The main area is divided into two sections: 'create a new template' and 'view an existing template'. The 'create a new template' section includes the instruction 'Drag and drop a file here or click to select a file'. The 'view an existing template' section contains a table of existing templates.

E-Signature Templates			
1040EZ	Edit	Sign	Delete
Application to Rent	Edit	Sign	Delete
David acf 1	Edit	Sign	Delete
Form I-9	Edit	Sign	Delete
Medical Records Release	Edit	Sign	Delete
Tech Writing Sample	Edit	Sign	Delete

2. The document opens in the E-Signature Viewer:

☰ Checklist

- f1\_01\_0[0]
- f1\_02\_0[0]
- c1\_01\_0\_0[0]
- c1\_01\_0\_1[1]
- f1\_03\_0[0]
- f1\_04\_0[0]
- f1\_05\_0[0]
- f1\_06\_0[0]
- f1\_07\_0[0]
- f1\_08\_0[0]
- f1\_09\_0[0]
- f1\_10\_0[0]
- f1\_11\_0[0]
- f1\_12\_0[0]
- f1\_13\_0[0]
- f1\_14\_0[0]

**Form 720-CS**  
(Rev. September 2016)  
Department of the Treasury  
Internal Revenue Service

**Carrier Summary Report**

For the month ending September, 2016.

OMB No. 1545-1733

Corrected  Void

**Part I Carrier**

Company name <u>ABC Company</u>		Employer identification number (EIN)	
Address (street, apt., room or suite number) <u>123 West Main Street</u>		Form 637 registration number	
City, state, and ZIP code (Foreign addresses, include province and postal code as appropriate. Do not abbreviate country name.) <u>Anywhere, NM 12345</u>			
Contact person <u>Ms. Smith</u>	Daytime telephone number <u>800-456-3456</u>	Fax number	Email address <u>msmith@abccompany.com</u>

**Part II Transactions for the Month**

	Net Gallons (attach additional schedule(s) if needed)			
	(a)	(b)	(c)	(d)
PC:	PC:	PC:	PC:	
<b>1 Total receipts.</b> Enter the total net gallons from Schedules A, column (g), by PC. If you have receipts from more than one facility for a PC, add the amounts from each facility's Schedule A and enter the combined total by PC.				
<b>2 Total deliveries.</b> Enter the total net gallons from Schedules B, column (g), by PC. If you have deliveries to more than one facility for a PC, you must add the amounts from each facility's Schedule B and enter the combined total by PC.				

Under penalties of perjury, I declare that I have examined this return, including accompanying schedules and statements, and to the best of my knowledge and belief, it is true, correct, and complete.

Signature ▶ \_\_\_\_\_ Title, if applicable ▶ \_\_\_\_\_ Date ▶ \_\_\_\_\_

Ms. Smith  
Type or print your name below signature.

For Privacy Act and Paperwork Reduction Act Notice, see the separate instructions. Cat. No. 720791 Form 720-CS (Rev. 9-2016)

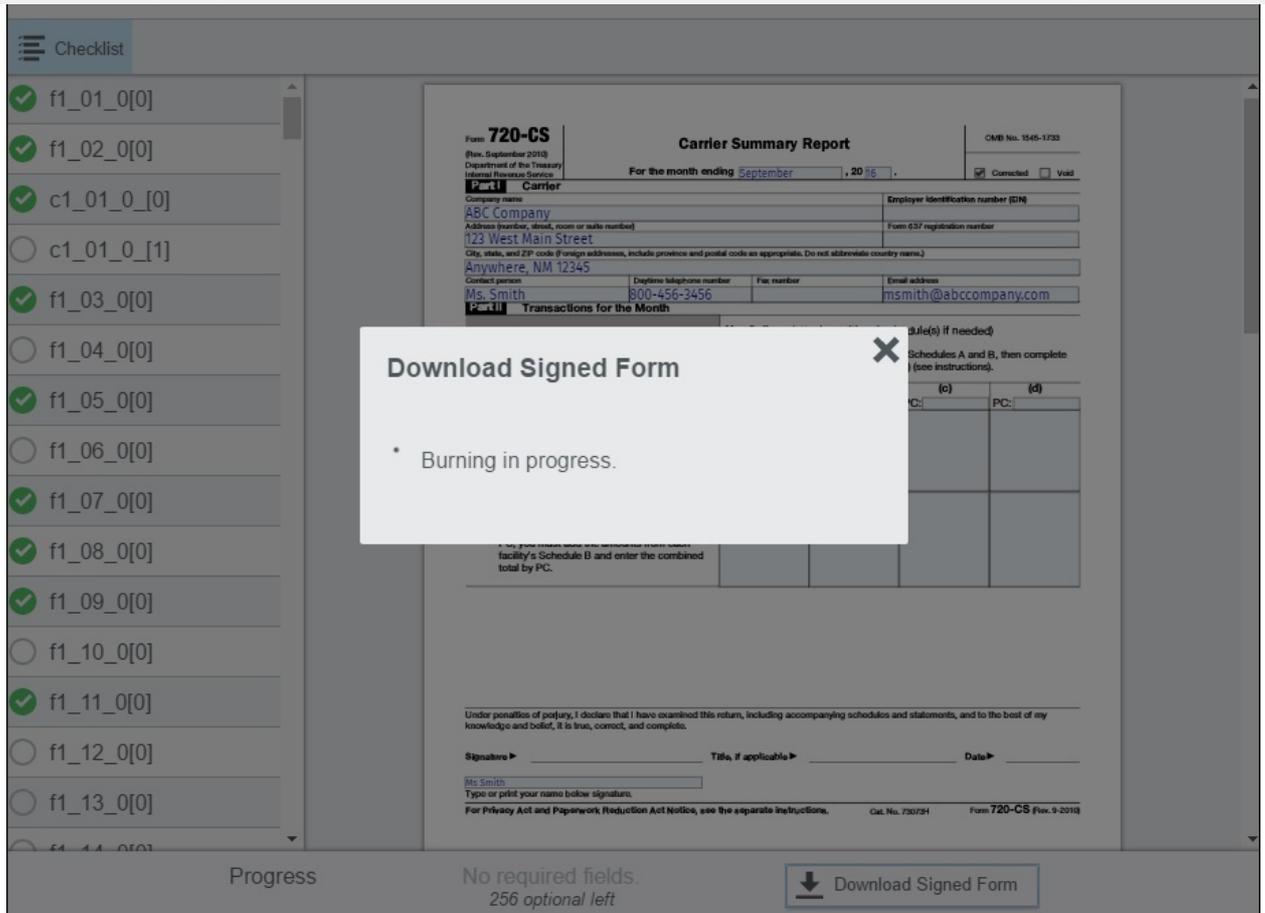
Progress

No required fields.  
256 optional left

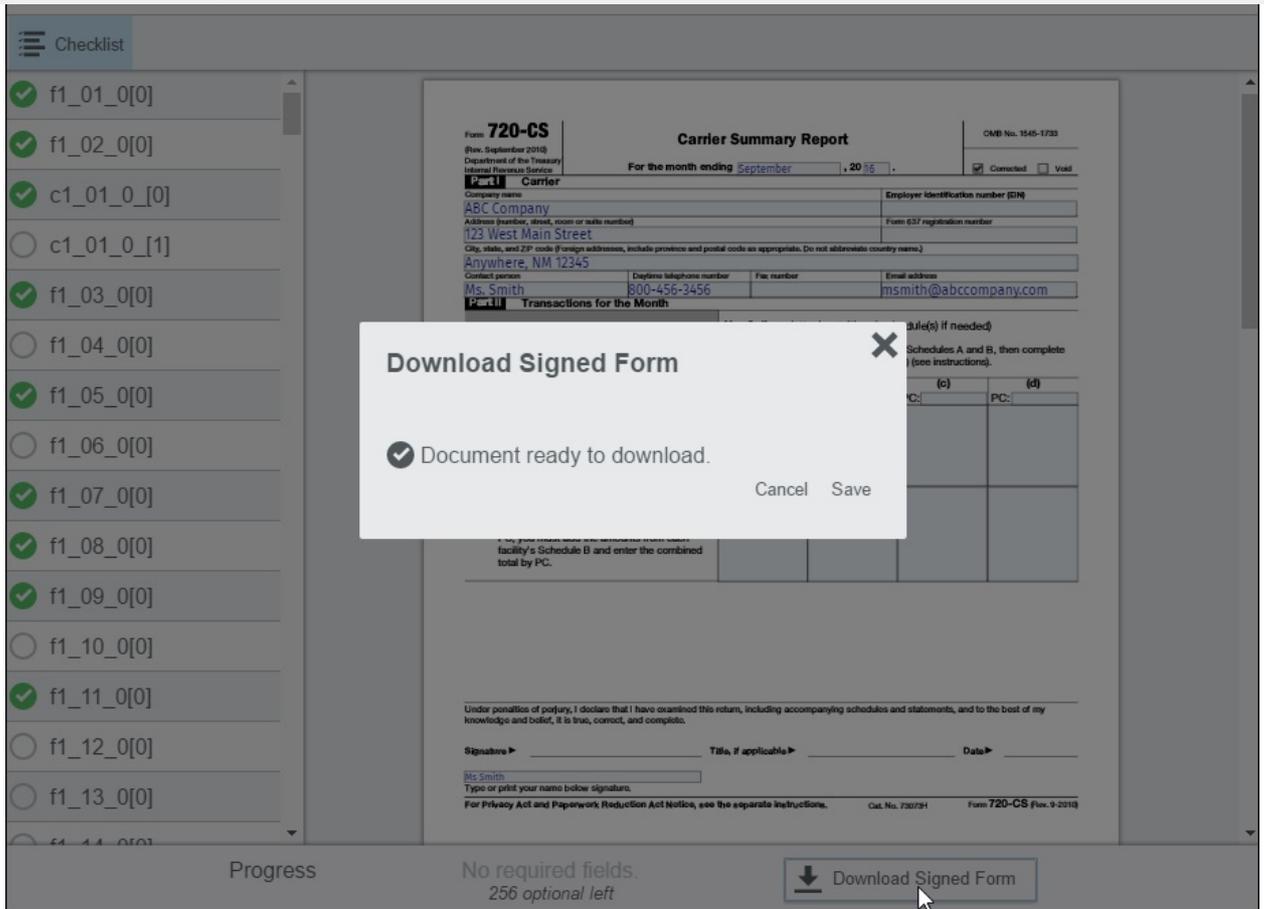
Download Signed Form

Fill out the fields and type in your signature at the bottom of the document.

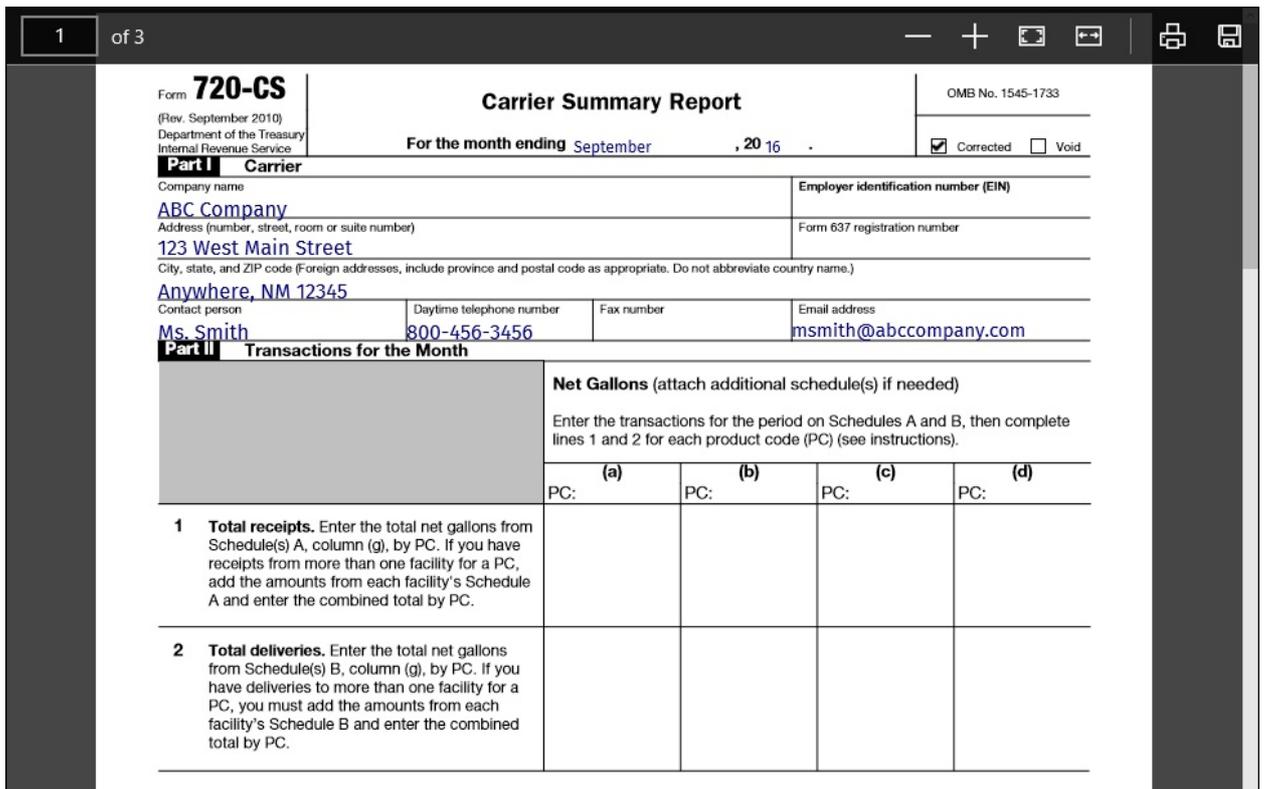
- When you are finished, click **Download Signed Form** and the Download Signed Form dialog displays:



4. Once the document is ready to be downloaded, the Download Signed Form dialog displays. Click **Save**:



5. You can open the downloaded document to view and share:



## Local File Viewer Guide

### Overview

Local File Viewer is a Windows application that works in conjunction with PrizmDoc to allow you to open files on your local machine in a web browser. Through the configuration application or by configuring command line parameters in the installation program, the Local File Viewer can be registered as the default viewer for a variety of files types, including PDF, Office files, JPG, TIFF, and several AutoCAD formats. This can be a useful alternative or addition to running the Viewer as a zero footprint web browser. For additional details, refer to the [Local File Viewer Help File](#) on Accusoft.com.

### Differences with the Viewer

While based on the latest version of the Viewer that ships with PrizmDoc, there are a few differences. The Local File Viewer supports most of the functionality available in the Viewer, with the exception of annotations and redactions. Annotations and redactions will be added in a future release. In addition, the Viewer API's and customization options supported in PrizmDoc are not currently supported in the Local File Viewer.

### Requirements

A web server with PrizmDoc installed is required to be running on a machine that has PrizmDoc installed in order for Local File Viewer to function. This can be either installed on the user's local machine (suitable for offline file viewing), or on a network (typical PrizmDoc installation). Details can be found in the [Local File Viewer Help File](#) on Accusoft.com.

## API Reference

This section contains the following information:

- [PrizmDoc Viewer API](#)
- [PrizmDoc Application Services \(PAS\) RESTful API](#)
- [PrizmDoc Server RESTful API](#)
- [PrizmDoc E-Signature Viewer API](#)

## PrizmDoc Viewer API

### Overview

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.
- 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.



You may want to review the section, [Configuring the Viewer](#) for additional information.

This section contains the following information:

- [API Functionality](#)
- [Getting Started](#)
- [Learning More](#)

### API Functionality

Functionality that is exposed through the Viewer API includes:

- Creating and destroying the Viewer
- Events

- 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

Basic knowledge for getting started using the API is covered below.

### 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.
```

## 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 un-responsive 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

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
```

```
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})  
    .setFillColor("#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) {
```

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.

## Learning More

To learn about the various areas of API functionality, examples and an API reference are provided:

- **API Examples** - The API examples demonstrate using the API to implement common stories.
- **API Reference** - The API reference covers the details of all API members.

## HTML Templates

This object contains the templates used in the Viewer:

Parameter	Data Type	Template Name	Description
comment	String	commentTemplate.html	The Comment template
contextMenu	String	contextMenuTemplate.html	The Context Menu template
copyOverlay	String	copyOverlayTemplate.html	The Copy Selected Text Modal template
downloadOverlay	String	downloadOverlayTemplate.html	The Download Modal template
esignOverlay	String	esignOverlayTemplate.html	The E-Signature Modal template, which is used for managing e-signatures, creating freehand e-signatures, and creating freehand text e-signatures
hyperlinkMenu	String	hyperlinkMenuTemplate.html	The Hyperlink Menu template
imageStampOverlay	String	imageStampOverlayTemplate.html	The Select Image Stamp

overwriteOverlay	String	overwriteOverlayTemplate.html	The Overwrite Existing Annotation File Modal template
pageRedactionOverlay	String	pageRedactionOverlayTemplate.html	The Redact Full Pages Modal template
print	String	printTemplate.html	The Print template
printOverlay	String	printOverlayTemplate.html	The Print Options Modal template
unsavedChangesOverlay	String	unsavedChangesOverlayTemplate.html	The Unsaved Annotation Changes Modal template
viewer	String	viewerTemplate.html	The Viewer Navigation and Layout template

## Developer Reference

This section contains the following information:

- PrizmDoc Viewer API
  - [Overview](#)
  - [HTML Templates](#)
  - [Developer Reference](#)
  - [External: jQuery](#)
  - [Global](#)
  - [Class: AjaxResponse](#)
  - [Class: BurnRequest](#)
  - [Class: Comment](#)
  - [Class: Conversation](#)
  - [Class: ConversionRequest](#)
  - [Class: DocumentHyperlink](#)
  - [Class: Error](#)

- Class: ImageStamps
- Class: LoadMarkupLayersRequest
- Class: Mark
- Class: MarkupLayer
- Class: MarkupLayerCollection
- Class: MouseTool
- Class: ObservableCollection
- Class: PrintRequest
- Class: Promise
- Class: SearchRequest
- Class: SearchResult
- Class: SearchTask
- Class: SearchTaskResult
- Mixin: SessionData
- Class: SignatureControl
- Class: SignatureDisplay
- Class: ThumbnailControl
- Class: Viewer
- Class: ViewerControl
- Mixin: Data
- Namespace: Ajax
- Namespace: fn
- Namespace: Language
- Namespace: MarkSchema
- Namespace: MarkupLayerSchema
- Namespace: MouseTools
- Namespace: PCCViewer
- Namespace: Signatures
- Namespace: Util

## External: jQuery

### External: jQuery

#### jQuery

The jQuery Plugin namespace.

See: [jQuery Plugins](#)

fn

---

Documentation generated by [JSDoc 3.3.3](#) on Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Global

### Global

#### Members

##### `convertedDocumentDownloadURL`

Deprecated since v11.0 (use the [PCCViewer.ConversionRequest#convertedDocumentDownloadURLs](#) property instead).

---

Documentation generated by [JSDoc 3.3.3](#) on Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Class: AjaxResponse

### 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:**

response Object

### *Properties*

Name	Type	Description
headers	Object	
status	Number	
statusText	String	
responseText	String	

### *Properties:*

Name	Type	Description
status	Number	
statusText	String	
responseText	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 Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Class: BurnRequest

### 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 onSuccessulBurn(burnturl)  
{  
    alert("burntURL = " + burnturl);  
    console.log(burnturl);  
}
```



BurnProgress	string	Triggered when the burn process receives an update.  Augmented properties of the <b>PCCViewer.Event</b> object for this event: <ul style="list-style-type: none"><li>• percent {number} - Indicates the estimated percentage complete.</li></ul>
BurnCompleted	string	Triggered when the burn process completes successfully, fails to complete, or is cancelled.  Augmented properties of the <b>PCCViewer.Event</b> object for this event: None
BurnFailed	string	Triggered when the burn process fails.  Augmented properties of the <b>PCCViewer.Event</b> object for this event: None
BurnCancelled	string	Triggered when user cancels burn process.  Augmented properties of the <b>PCCViewer.Event</b> object for this event: None
BurnWorkerCreated	string	Triggered when the burn process is created by the back-end burn service.  Augmented properties of the <b>PCCViewer.Event</b> 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.*

- 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

(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

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.getBurnedDocumentDownloadU
```

`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

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

```
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}

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.Bu

    function(ev) {
        alert("Document burn
completed.");
    })

.on(PCCViewer.BurnRequest.EventType.Bu
```

```
        event.percent + "%");  
    });  
});
```

`then(onFulfilledopt, onRejectedopt)` → `{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
onFulfilled	<code>PCCViewer.Promise~onFulfilled</code>	<optional>	Called if or when the promise is resolved. Optionally pass a value of null or undefined if you do not use this callback, but you want to provide an onRejected callback.
onRejected	<code>PCCViewer.Promise~onRejected</code>	<optional>	Called if or when the promise is rejected.

**Returns:**

A promise object that is resolved according to the Promises/A+ standard.

Type

## Example

```
var viewerControl =
$("#myElement").pccViewer(...).viewerC

// 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 Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Class: Comment

### 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

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".
```

```
// 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
KVPs are stored.
comment.getDataKeys(); // returns []

// Returns a list of all keys.
comment.setData("Author", "Mark");
comment.setData("Severity",
"Critical");
comment.getDataKeys(); // returns
["Author", "Severity"]
```

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.

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>

```
var comment =
myConversation.addComment("Hello
world.");

// Returns an empty array before
KVPs 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 Date Object or a string in the ISO 8601 format.

See: [PCCViewer.Comment#getCreationTime](#)

- 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 KVPs 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:**

key string The key for which to set the data value.

value string This is the value to set for the key.

- 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);
```

```
// 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"><li>• This must be a string or undefined.</li><li>• The maximum length of the string is not limited by this function.</li></ul>

See: [PCCViewer.Data#setSessionData](#)  
[PCCViewer.Comment#getSessionData](#)  
[PCCViewer.Comment#getSessionDataKeys](#)

#### Returns:

The Comment object on which the method was called.

Type

[PCCViewer.Comment](#)

#### Example

```
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:**

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 Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Class: Conversation

### 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	<a href="#">PCCViewer.Mark</a>	The Mark to which the conversation is attached.

**Throws:**

If mark is not a [PCCViewer.Mark](#) Object.

Type

Error

```
// 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
commentText	string	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:

comments	<a href="#">PCCViewer.Comment</a>   Array. < <a href="#">PCCViewer.Comment</a> >	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](#)

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](#)

### 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
KVPs 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.

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",
```

```
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  
KVPs 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:

given key.

- There is no artificial limit imposed on the number of key-value pairs that are stored.
- If limits on the number of KVPs 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"><li>• This must be a string or undefined.</li><li>• The maximum length of the string is not limited by this function.</li></ul>

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",
```

```
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"><li>• This must be a string or undefined.</li><li>• The maximum length of the string is not limited by this function.</li></ul>

[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
```

```
    } catch (e) { // throws  
        conversation.setSessionData("FooBar",  
        {}); // throws  
        conversation.setSessionData("FooBar",  
        []); // throws  
    }  
}
```

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Class: ConversionRequest

### 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

```
onSuccessfulConvert(convertedDocuments)
{
    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(onSuccessfulCon
onFailedConvert);

//register some events
conversionRequest

.on(PCCViewer.ConversionRequest.EventT

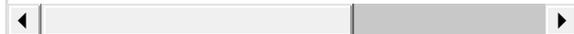
    function(ev) {
        alert("Document conversion
completed.");
    })

.on(PCCViewer.ConversionRequest.EventT

    function(event) {
        alert("Conversion
progress: " + event.percent + "%");
    })

.on(PCCViewer.ConversionRequest.EventT

    function(event) {
        alert("Document conversion
failed.");
    });
```



Members

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	Type	Description
ConversionProgress	string	Triggered when the conversion process receives an update.  Augmented properties of the <code>PCCViewer.Event</code> object for this event: <ul style="list-style-type: none"><li>• progress {number} - Indicates the estimated percentage complete.</li></ul>
ConversionCompleted	string	Triggered when the conversion process completes successfully, fails to complete, or is cancelled.  Augmented properties of the <code>PCCViewer.Event</code> object for this event: None
ConversionFailed	string	Triggered when the conversion process fails.  Augmented properties of the <code>PCCViewer.Event</code> object for this event: None
ConversionCancelled	string	Triggered when user cancels the conversion process.  Augmented properties of the <code>PCCViewer.Event</code> object for this event:

	None
ConversionWorkerCreated	string
	Triggered when the conversion process is created by the back-end conversion service.
	Augmented properties of the <a href="#">PCCViewer.Event</a> 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 downloadUrIs =  
conversionRequest.convertedDocumentDow
```

**(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*

*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

## 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

```
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.getConvertedDocument
```

`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

### 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:**

eventType	string	A string specifying the event type. See <a href="#">PCCViewer.ConversionRequest.EventType</a> for a list and description of all supported events.
handler	<a href="#">PCCViewer.Event~eventHandler</a>	A function that was attached previously to the ViewerControl.  <b>Note:</b> This must be the same function object previously passed to <a href="#">PCCViewer.ConversionRequest#on</a> . 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
eventType	string	A string that specifies the event type. This value is case-insensitive. See <a href="#">PCCViewer.ConversionRequest.EventType</a> for a list and description of all supported events.
handler	<a href="#">PCCViewer.Event~eventHandler</a>	A function that will be called whenever the event is triggered.

See: [PCCViewer.ConversionRequest#off](#)  
[PCCViewer.ViewerControl#on](#) for more details on event subscription.

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
<code>onFulfilled</code>	<code>PCCViewer.Promise~onFulfilled</code>	<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>	<code>PCCViewer.Promise~onRejected</code>	<optional>	Called if or when the promise is rejected.

#### Returns:

A promise object that is resolved according to the

Type

PCCViewer.Promise

### Example

```
var viewerControl =
$("#myElement").pccViewer(...).viewerC

// a basic example
viewerControl.requestDocumentConversio

    function onFulfilled(url) {
        // download document using
`url`
    },
    function onRejected(error) {
        // Handle failure
    }
);
```

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Class: DocumentHyperlink

### 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:

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.
  - Their attributes href, position, and styling cannot be modified.
  - They cannot be deleted.
  - 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 Viewer.

### 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

viewerControl.requestDocumentHyperlinks

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":
```

```
        // intercept the click 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](#)

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();
```

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 Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Class: Error

### 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 Viewer and Server, the code should be PascalCased and should not contain any spaces.
message	string	A developer readable string that indicates details of the error.

```
// 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 Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Class: Event

### 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	<a href="#">PCCViewer.ViewerControl</a>	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.PageCoun
pageCountReadyHandler);
```

**getType()** → {string}

as the event type 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.PageCoun
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	<code>PCCViewer.Event</code>	A <code>PCCViewer.Event</code> 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#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 Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Class: ImageStamps

### 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.

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

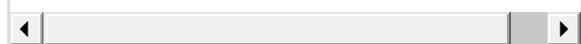
#### Parameters:

Name	Type	Description
imageStampId	string	

### Example

```
var options = { imageHandlerUrl:
  "../pcc.ashx" }
var stampApi = new
  PCCViewer.ImageStamps(options);

var imageUrl =
  stampApi.getImageSourceURL(imageStampI
```



#### requestImageSourceBase64(imageStampId) →

{PCCViewer.Promise}

Retrieves the dataUrl (base64 image source), and dataHash (unique id) from the server for the provided imagesStampId

If unable to retrieve image stamp data URL and dataHash 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 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

a `PCCViewer.Error` object with its `code` property set to `Error`.

**Parameters:**

Name	Type	Description
<code>imageStampId</code>	<code>string</code>	

**Returns:**

a Promise object, on success returns a JSON object, on failure promise is rejected with reason. Returned JSON object will have `dataUrl`, and `dataHash` properties: `dataUrl`: base64 encoded image source `dataHash`: Encoded unique Id

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(image

    base64Promise.then(
      function
onSuccess(imageSource) {
        base64Source =
imageSource

        var pageNumber = 1;
        var rectangle = {
x: 30, y: 30, width: 100, height:
100 };

        var mark =
```

```
imageStampAnnotation },
                mark.setImage({
                    dataUrl:
base64Source.dataUrl, id:
base64Source.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 <code>imageHandlerUrl</code> property.

**Returns:**

a Promise object.

Type

**PCCViewer.Promise**

**Example**

```
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(image

    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(pageNumbe
"ImageStampAnnotation");
        mark.setImage({
          dataUrl:
base64Source.dataUrl, id:
base64Source.dataHash
        });

mark.setRectangle(rectangle);
      }
    );

  },
  function onFailure(error) {
    alert(error.message ?
error.message : error)
  }
);
```

## Class: LoadMarkupLayersRequest

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
viewerControl	string	The <a href="#">PCCViewer.ViewerControl</a> for the loaded document.
layerRecordIds	Array. <string>	An array of layer record IDs.

### Example

```
function  
onSuccessfulLoad(annotationLayers) {  
    console.log(annotationLayers);  
}  
  
function onFailedConvert(error) {  
    alert("Markup layer record loading
```

```
error.message : error));
}

// A LoadMarkupLayersRequest object
is created by and returned from the
call to the
ViewerControl#convertDocument method
var LoadMarkupLayersRequest =
viewerControl.loadMarkupLayers(['abc12

LoadMarkupLayersRequest.then(onSuccess
onFailedConvert);

//register some events
LoadMarkupLayersRequest

.on(PCCViewer.LoadMarkupLayersRequest.

    function(ev) {
        alert("Markup layer
loading completed.");
    })

.on(PCCViewer.LoadMarkupLayersRequest.

    function(event) {
        alert("Conversion
progress: " + event.percent + "%");
    })

.on(PCCViewer.LoadMarkupLayersRequest.

    function(event) {
        alert("Markup layer
loading failed.");
    })

.on(PCCViewer.LoadMarkupLayersRequest.

    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.

Instead of using this enumeration, you can pass string values of the eventType (enumeration values)

**Type:**

- string

**Properties:**

Name	Type	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.

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`.

The cancelled request.

Type

`PCCViewer.LoadMarkupLayersRequest`

### Example

```
var loadMarkupLayersRequest =  
viewerControl.loadMarkupLayers(['abc12  
  
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(['abc12  
  
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(['abc12

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(['abc12

var viewerControl =
loadMarkupLayersRequest.getViewerContr
```

`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 <code>PCCViewer.LoadMarkupLayersRequest.EventType</code> for a list and description of all supported events.
handler	<code>PCCViewer.Event~eventHandler</code>	A function that was attached previously to the <code>ViewerControl</code> .  <b>Note:</b> This must be the same function object previously passed to <code>PCCViewer.LoadMarkupLayersRequest#on</code> . 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
<code>eventType</code>	<code>string</code>	A string that specifies the event type. This value is case-insensitive. See <a href="#">PCCViewer.LoadMarkupLayersRequest.EventType</a> for a list and description of all supported events.
<code>handler</code>	<a href="#">PCCViewer.Event~eventHandler</a>	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](#)

## Class: Mark

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

.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 programatically 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  
comaptible 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

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"><li>• type {string} [required] Mark Type <a href="#">PCCViewer.Mark.Type</a></li><li>• pageNumber {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.</li></ul>

See: [PCCViewer.ViewerControl#addMark](#)  
[PCCViewer.MouseTool#getTemplateMark](#)  
[PCCViewer.Mark.Type](#)

#### Example

```
var viewerControl = new
PCCViewer.ViewerControl(...);

// use
PCCViewer.ViewerControl#addMark(pageN
markType) to create
// and add a mark instead of new
Mark()
viewerControl.addMark(1,
"LineAnnotation");
```

(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	Type	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.Fon

// 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:**

### Properties:

Name	Type	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

// 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	Type	Description
Full	string	Specifies that the mark is fully interactive using the mouse, touch-input, and API.
SelectionDisabled	string	Specifies that the mark cannot be selected.  <b>Note:</b> 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.Mar

// or just use the string value of
the enumeration
mark.setInteractionMode("SelectionDis
```



(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	Type	Description
None	string	No line head.
FilledTriangle	string	A filled triangle line head. <b>Note:</b> this makes the line an arrow. :)

### Example

```
// use the enumeration to make a
line an arrow
```

```
// or just use the string value  
myLineAnnotation.setEndHeadType("Fill
```

(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	Type	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

		<p>text on the document. The text has a background and border.</p> <p>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>.</p> <p>The methods specific for this are: <code>highlightText</code>.</p>
<code>StampAnnotation</code>	<code>string</code>	<p>This mark is drawn as a stamp on the document. A stamp has a label (text) and a border.</p> <p>The properties specific for this are: <code>color</code>, <code>label</code>, <code>rectangle</code>.</p>
<code>HighlightAnnotation</code>	<code>string</code>	<p>This mark is drawn as a text highlight on the document.</p> <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>
<code>RectangleRedaction</code>	<code>string</code>	<p>This mark is drawn as a rectangle on the document.</p> <p>The properties specific for this are: <code>rectangle</code>.</p>
<code>TransparentRectangleRedaction</code>	<code>string</code>	<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>

		The properties specific for this are: <code>rectangle</code> .
<code>TextHyperlinkAnnotation</code>	<code>string</code>	<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>
<code>TextRedaction</code>	<code>string</code>	<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>
<code>StampRedaction</code>	<code>string</code>	<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>
<code>FreehandAnnotation</code>	<code>string</code>	<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>
<code>FreehandSignature</code>	<code>string</code>	This mark is drawn as a signature on the

		<p>document and is confined to a rectangle.</p> <p>The properties specific for this are: path, rectangle, color, thickness.</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: text, rectangle, color, fontName.</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 the mouse, touch-screen, and keyboard in order to set the text and adjust the rectangle.</p> <p>The properties specific for this are: text, rectangle, fontColor, fontName, mask, horizontalAlignment, maxLength.</p>
TextAreaSignature	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: text, rectangle, fontColor, fontName, maxFontSize, fontStyle,</p>

		horizontalAlignment, maxLength.
TextSelectionRedaction	string	<p>This mark is drawn as a text highlight redaction on the document.</p> <p>The properties specific for this are: text.</p> <p>The methods specific for this are: highlightText.</p>
ImageStampAnnotation	string	<p>This mark is drawn as an ImageStamp annotation on the document.</p> <p>The properties specific for this are: none.</p>
ImageStampRedaction	string	<p>This mark is drawn as an ImageStamp redaction on the document.</p> <p>The properties specific for this are: none.</p>
PolylineAnnotation	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: color, thickness, opacity, points.</p>
StrikethroughAnnotation	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: color, thickness and text.</p> <p>The methods specific for this are:</p>

highlightText.

### 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:

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#getBoundingRectangle](#)

**Example**

```
var boundingRectangle =  
mark.boundingRectangle;
```

**color :string**

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.

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

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#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

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.Select
```

**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*

**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.

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

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](#)

```
// 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
{
    // 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.*

**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.";
}
```

## 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 PPlaceSignature 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

## 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.*

- string

See: [PCCViewer.Mark.Type](#)  
[PCCViewer.Mark#getType](#)

### Example

```
switch (mark.type) {
  case
  PCCViewer.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](#).

[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:

**Returns:**

A rectangle object of the type {x: xValue, y: yValue, width: widthValue, height: heightValue}.

Type

Object

**Example**

```
var boundingRectangle =  
mark.getBoundingBox();
```

**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:**

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");
```

```
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
// KVPs 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
```

```
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**

```
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**

```
var fontStyle =  
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](#)

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;
```

```
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

```
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

```
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**

```
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) {
```

```
    ]
```

**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

```
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](#)

**Returns:**

The reason contained in the Redaction mark.

Type  
string

**Example**

```
if (mark.getReason) {  
    var reason = mark.getReason();  
}
```

**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](#)

A rectangle object of the type {x: xValue, y: yValue, width: widthValue, height: heightValue}.

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

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
KVPs 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('Ac

// get the template mark
var signatureTemplateMark =
accusoftPlaceSignature.getTemplateMar
```

```
// 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.

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](#)

The type of Mark.

Type

string

### Example

```
switch (mark.getType()) {
    case
PCCViewer.Mark.Type.LineAnnotation:
    ...
    break;
    default:
    ...
}
```

**setVisible()** → {boolean}

Gets the border color 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

This method is defined on marks of type: TextAnnotation, TextRedaction, HighlightAnnotation, TextSelectionRedaction, TextHyperlinkAnnotation and StrikethroughAnnotation.

***Parameters:***

`highlights` Array.  
<object>  
| object

An array of objects or a single object that defines a highlight.

- Each object has the following properties:
  - `startIndex` {number} - required
    - The start index of the highlight in the mark text.
    - The valid range is  $0 \leq \text{startIndex} < \text{mark.getText().length}$ .
  - `length` {number} - required
    - 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  $\text{length} > 0$ .
  - `color` {string} - required
    - 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} - optional
    - Specifies the opacity of the highlight.
    - Valid values are from 0 to 255 (inclusive).
    - If unspecified, an opacity value of 127 will be used.

If passed a value of null, undefined, or an

empty array, then the highlights are cleared.

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}
    ]);
}
```

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](#)

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 KVPs 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"><li>• This must be a string or undefined.</li><li>• The maximum length of the string is not limited by this function.</li></ul>

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
```

```
// 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	<a href="#">PCCViewer.Mark.LineHeadType</a>	The line head type. For example, <a href="#">PCCViewer.Mark.LineHeadType.FilledRectangle</a> 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.Li
}
}
```

**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**

```
mark.setFillFillColor({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.setFillFillColor) {  
    mark.setFillFillColor("#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:**

**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

```
mark.setFontSize(12);  
}
```

**setFontStyle(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:**

value Array. An array containing values containing the <string> font styles for the text in the text annotation.

Valid values in the array are:

- "Bold"  
(PCCViewer.Mark.FontStyles.Bold)
- "Italic"  
(PCCViewer.Mark.FontStyles.Italic)
- "Underline"  
(PCCViewer.Mark.FontStyles.Underline)
- "Strikeout"  
(PCCViewer.Mark.FontStyles.Strikeout)

**Note:** An empty array would render the text with normal font style.

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:

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	<a href="#">PCCViewer.Mark~ImageData</a>	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

### 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 <b>PCCViewer.Mark.InteractionMode</b> , 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.Mar
```

**setLabel(value)** → **{PCCViewer.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:**

mask object The mask to set on this mark to assist user input.

### Properties

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](#)

```
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
value	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); //
    translucent
    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:**

**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

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`,

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.

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.HighlightAnnotati

        .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
```

```
pages.  
// We use a helper method to ensure  
that the ViewerControl has text for  
all pages that it will span.  
ensureViewerControlHasPageText(viewer  
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.HighlightAnnotati  
  
        .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(viewer  
markPageNumber, markPostion) {  
  
    // Calling
```

```
cause the viewer control 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(markPag

    // 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
```

```
    if (extendsToNextPage)
    {
        if (markPageNumber
        < viewerControl.getPageCount()) {
            return
            ensureViewerControlHasPageText(viewer
            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](#)

**Returns:**

The Mark object on which this method was called.

Type  
[PCCViewer.Mark](#)

```
if (mark.setReason) {
    mark.setReason("Information for
top security clearance only.");
}
```

**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: {x: xValue, y: yValue, width: widthValue, height: heightValue}

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}**

[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"><li>• This must be a string or undefined.</li><li>• The maximum length of the string is not limited by this function.</li></ul>

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",
```

```
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 undefined will reset the state of the mouse tool back to default.  See <a href="#">PCCViewer.Signatures~FreehandSignature</a> and <a href="#">PCCViewer.Signatures~TextSignature</a>

See: [PCCViewer.Mark#getSignature](#)  
[PCCViewer.Mark#signature](#)

**Throws:**

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('Ac

// get the template mark
var signatureTemplateMark =
accusoftPlaceSignature.getTemplateMar

// set signature path for freehand
signature
signatureTemplateMark.setSignature({p
"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:

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.

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:

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	Type	Description
id	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.
dataUrl	string	A base64-encoded data URL representation of an image.

## Class: MarkupLayer

Class: MarkupLayer

PCCViewer.MarkupLayer

(protected) new MarkupLayer(viewerControl, markReferences<sub>opt</sub>)

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
viewerControl	string		The <a href="#">PCCViewer.ViewerControl</a> for the loaded document.
markReferences	Object   Array. <Object>	<optional>	The JSON reference node (or an array of them) from the markup layer record.

### Example

```
// Optionally, specify any references to marks on another layer
```

```
var markReference = {
    creationDateTime: "2015-06-
12T21:20:58.527Z"
    data: {
        "key1": "value1",
        "key2": "value2"
    },
    markUid:
"ZZZrOV8yMDE1LTA2LTExVDE50jU50jEwLjE3M

    text: "user 1 comment on user 3
mark"
};

// Create a new layer
var layer = new
PCCViewer.MarkupLayer(viewerControl,
markReference)
// 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())

//register some events
layer

.on(PCCViewer.MarkupLayer.EventType.Ma

    function(ev) {
        alert("Markup layer
created.");
    })

.on(PCCViewer.MarkupLayer.EventType.Ma

    function(ev) {
        alert("Markup layer
destroyed.");
    })

.on(PCCViewer.MarkupLayer.EventType.Ma

    function(event) {
        alert("Mark added to
layer: " + ev.marks[0].getId());
```

```
.on(PCCViewer.MarkupLayer.EventType.Ma

    function(event) {
        alert("Mark removed from
layer: " + ev.marks[0].getId());
    })

.on(PCCViewer.MarkupLayer.EventType.Ma

    function(event) {
        alert("Layer's interaction
mode changed to: " +
ev.interactionMode);
    })

.on(PCCViewer.MarkupLayer.EventType.Ma

    function(event) {
        alert("MarkupLayer#show()
called.");
    })

.on(PCCViewer.MarkupLayer.EventType.Ma

    function(event) {
        alert("MarkupLayer#hide()
called.");
    });
```

## 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	Type	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

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	<a href="#">PCCViewer.Mark</a>   Array. < <a href="#">PCCViewer.Mark</a> >	{ <a href="#">PCCViewer.Mark</a> } object or an array of them.

**Returns:**

The markup layer Object.

Type

[PCCViewer.MarkupLayer](#)

**Example**

```
var markupLayer =  
viewerControl.getMarkupLayerCollection  
[0]; // Get the first markup layer.  
var mark = viewerControl.addMark(1,  
'HighlightAnnotation'); // Create a  
new mark  
markupLayer.addMarks(mark);
```

Copies marks from other layers to this layer.

**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
[0]; // Get the first markup layer.
var markupLayer2 =
viewerControl.getMarkupLayerCollection
[1]; // Get the second markup layer.
var markupLayer4 =
viewerControl.getMarkupLayerCollection
[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  
[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.

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
KVPs are stored.
layer.getDataKeys(); // returns []

// Returns a list of all keys.
layer.setData("Username", "Admin");
```

```
    layer.getDataKeys(); // returns  
    ["Username", "CreatedDate"]
```

**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  
[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  
[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  
[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  
[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  
[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 "CreateDate" is set the value "1970-01-01".
layer.setSessionData("CreateDate", "1970-01-01");

layer.getSessionData("Username"); // returns "Admin"
layer.getSessionData();           // returns {"Username":"Admin", "CreateDate":"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:**

no data is stored, then an empty array will be returned.

Type

Array.<string>

### Example

```
// Returns an empty array before
KVPs 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
[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
[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
[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:**

eventType	string	A string specifying the event type. See <a href="#">PCCViewer.MarkupLayer.EventType</a> for a list and description of all supported events.
handler	<a href="#">PCCViewer.Event~eventHandler</a>	A function that was attached previously to the <code>ViewerControl</code> . <b>Note:</b> This must be the same function object previously passed to <a href="#">PCCViewer.MarkupLayer#on</a> . 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 <a href="#">PCCViewer.MarkupLayer.EventType</a> for a list and description of all supported events.
handler	<a href="#">PCCViewer.Event~eventHandler</a>	A function that will be called whenever the event is triggered.

See: [PCCViewer.MarkupLayer#off](#)  
[PCCViewer.ViewerControl#on](#) for more details on event subscription.

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	<code>PCCViewer.Mark</code>   Array. < <code>PCCViewer.Mark</code> >	{ <code>PCCViewer.Mark</code> } 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 KVPs are required, they

- 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"><li>• This must be a string or undefined.</li><li>• The maximum length of the string is not limited by this function.</li></ul>

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"
```

```
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 <a href="#">PCCViewer.Mark.InteractionMode</a> ,

**Returns:**

The markup layer Object on which this method was called

Type

[PCCViewer.MarkupLayer](#)

**Example**

```
var markupLayer =  
viewerControl.getMarkupLayerCollection  
[0]; // Get the first markup layer.  
markupLayer.setInteractionMode(PCCView
```

**setName(name)** → {[PCCViewer.MarkupLayer](#)}

### 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  
[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  
[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"><li>• This must be a string or undefined.</li><li>• The maximum length of the string is not limited by this function.</li></ul>

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
```

```
var uc = 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
```

```
markupLayer.show(); // Show the
marks in the layer.
```

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:45 GMT-0400 (Eastern Daylight Time)

## Class: MarkupLayerCollection

### 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
viewerControl	string	The <a href="#">PCCViewer.ViewerControl</a> 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();
```

```
layerCollection.Remove(layers[id].GetItem());

//register some events
layerCollection

.on(PCCViewer.MarkupLayerCollection.EventType.Added)

    function(ev) {
        alert("Markup layer added
to the collection with id = " +
ev.layerId);
    })

.on(PCCViewer.MarkupLayerCollection.EventType.Removed)

    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	Type	Description
MarkupLayerAdded	string	
MarkupLayerRemoved	string	

See: [PCCViewer.Event](#)  
[PCCViewer.MarkupLayerCollection#on](#)  
[PCCViewer.MarkupLayerCollection#off](#)

(readonly) **viewerControl** :Object

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;
```

(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;
```

**Methods**

**addItem(layer)** →

{[PCCViewer.MarkupLayerCollection](#)}

This method is used to add a layer to the collection.

**Parameters:**

layer `PCCViewer.MarkupLayer` A markup layer object.

### 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, thisArgopt)` →  
{`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   <code>PCCViewer.MarkupLayerCollection~iterator</code>		The function to execute for each item in the collection.
thisArg	*	<optional>	The Object to be used as this for the iterator function.

### Throws:

If the iterator parameter is not a function.

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:

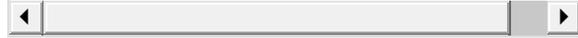
The number representing how many layers are in the collection.

Type

number

### Example

```
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:**

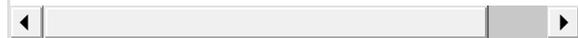
A markup layer object or undefined if layerId does not correspond to a layer in the collection.

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].getI
```



**getViewerControl()** → {**PCCViewer.ViewerControl**}

Gets the viewer control associated with this layer.

**Returns:**

A viewer control object.

Type

### 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
eventType	string	A string specifying the event type. See <a href="#">PCCViewer.MarkupLayerCollection.EventType</a> for a list and description of all supported events.
handler	<a href="#">PCCViewer.Event~eventHandler</a>	A function that was attached previously to the ViewerControl.  <b>Note:</b> This must be the same function object previously passed to <a href="#">PCCViewer.MarkupLayerCollection#on</a> . 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](#)

`{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 <a href="#">PCCViewer.MarkupLayerCollection.EventType</a> for a list and description of all supported events.
handler	<a href="#">PCCViewer.Event~eventHandler</a>	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].g
```

Documentation generated by [JSDoc 3.3.3](#) on Mon Mar 13 2017 16:48:45 GMT-0400 (Eastern Daylight Time)

## Class: MouseTool

### 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

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(n
type)
// function instead of this
constructor.
PCCViewer.MouseTools.createMouseTool("
"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](#).

- string

**Properties:**

Name	Type	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.
PlaceSignature	string	Use to click on the document and place a signature in that location.
TextSelectionRedaction	string	Use the TextSelectionRedaction mouse tool type to select text and create a text selection redaction.
ImageStampAnnotation	string	Use the ImageStampAnnotation mouse tool type to place ImageStamp annotation.
ImageStampRedaction	string	Use the ImageStampAnnotation mouse tool type to place ImageStamp redaction.
PolylineAnnotation	string	Use the PolylineAnnotation mouse tool type to place Polyline annotation.
StrikethroughAnnotation	string	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

## 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(mous

// Check if templateMark is a
property in the MouseTool object
if (templateMark in myMouseTool) {
    // get the template mark
```

```
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(mous  
  
// get the mouse tool's type  
var mouseToolType =  
myMouseTool.type;  
  
// do something with the type  
switch (mouseToolType) {  
    case  
PCCViewer.MouseTool.Type.LineAnnotatio  
  
        ...  
        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](#)

```
var myMouseTool =
PCCViewer.MouseTools.getMouseTool(mous

// 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(mous

// get the mouse tool's type
var mouseToolType =
myMouseTool.getType();

// do something with the type
```

```
case  
PCCViewer.MouseTool.Type.LineAnnotatio  
  
    ...  
    break;  
default:  
    ...  
}
```

Documentation generated by *JSDoc 3.3.3* on Wed Mar 29 2017 10:18:08 GMT-0400 (Eastern Daylight Time)

## Class: ObservableCollection

### Class: ObservableCollection

#### PCCViewer.ObservableCollection

**new ObservableCollection()**

Represents a dynamic collection that provides notifications when items get added or removed.

#### Members

(readonly) **EventType** :string

The known events for the collection.

#### Type:

- string

#### Properties:

Name	Type	Description
ItemAdded	string	Triggered when an item is added to the collection.
ItemRemoved	string	Triggered when an item is removed from the collection.

#### Methods

**add(item)** → {PCCViewer.ObservableCollection}

Add an item to the collection.

Name	Type	Description
item	*	Any object or value to add to the collection.

**Returns:**

The collection object.

Type

[PCCViewer.ObservableCollection](#)

`forEach(iterator, thisArgopt) → {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   <a href="#">PCCViewer.ObservableCollection~iterator</a>		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](#)

`{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}`

## 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`

## 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 Mon Mar 13 2017 16:48:45 GMT-0400 (Eastern Daylight Time)

## Class: PrintRequest

### 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:

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 <a href="#">PCCViewer.Event</a> object for this event:</p> <ul style="list-style-type: none"><li>• <code>index</code> {number} Indicates the index of the page that was prepared in respect to <code>totalPages</code>.</li><li>• <code>pageNumber</code> {number} Indicates the page number of the page that was prepared. This page number of the page in the document.</li><li>• <code>totalPages</code> {number} Indicates the total number of pages that are being printed.</li></ul>
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 <a href="#">PCCViewer.Event</a> object for this event:</p> <ul style="list-style-type: none"><li>• none</li></ul>
PrintCancelled	string	<p>Event triggered if printing is cancelled during the preparation process.</p> <p>Augmented properties of the <a href="#">PCCViewer.Event</a> object for this event:</p> <ul style="list-style-type: none"><li>• none</li></ul>

PrintFailed	string	Event triggered if the printing process failed due to an error.  Augmented properties of the <a href="#">PCCViewer.Event</a> object for this event: <ul style="list-style-type: none"><li>• none</li></ul>
-------------	--------	--

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

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.

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.P

    function(ev) {
        alert("Print completed.");
    })

.on(PCCViewer.PrintRequest.EventType.P

    function(ev) {
        alert("Print progress: " +
100 * (ev.index + 1) / ev.totalPages
+ "%");
    });
```

## Class: Promise

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 Viewer 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

Returns a promise that is fulfilled when all of the input promises are fulfilled. The returned promise is fulfilled with an array of the fulfillment 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"><li>• If the item is a PCCViewer.Promise, then the output value will be the fulfillment value of the promise.</li><li>• If the item is thenable, then the output value will be the fulfillment value of the thenable.</li><li>• Otherwise, the output value will be the item.</li></ul>

**Throws:**

If the promises argument is not an array.

Type  
TypeError

**Example**

```

    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(pageAttributePro

    function
    onFulfilled(allPageAttributes) {

    console.log(JSON.stringify(allPageAttr

    },
    function onRejected(error) {
        alert("Something went wrong
        getting the page attributes. " +
        (error.message ? error.message :
        error));
    }
    );

```

```
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(onFulfilledopt, onRejectedopt)` →  
`{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>	<code>PCCViewer.Promise~onFulfilled</code>	<optional>	Called if or when the promise is resolved. Optionally pass a value of null or undefined 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(...).viewerC

// 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,
```

```
);  
  
// 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 value 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
reason	*	The type and value of the reason 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 Mon Mar 13 2017 16:48:45 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:

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*

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

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

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:

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 <a href="#">PCCViewer.SearchRequest#on</a> for possible values.
handler	function	The function that was previously subscribed to the event type.

### Returns:

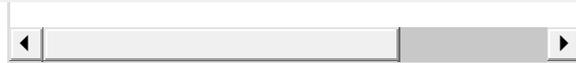
The SearchRequest object on which this method was called.

Type

[PCCViewer.SearchRequest](#)

### Example

```
// subscribe  
searchRequest.on(PCCViewer.EventType.S  
onSearchCompleted);  
  
// unsubscribe  
searchRequest.off(PCCViewer.EventType.  
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"><li>• "SearchCompleted" - <a href="#">PCCViewer.EventType.SearchCompleted</a></li><li>• "SearchFailed" - <a href="#">PCCViewer.EventType.SearchFailed</a></li><li>• "SearchCancelled" - <a href="#">PCCViewer.EventType.SearchCancelled</a></li><li>• "SearchResultsAvailable" - <a href="#">PCCViewer.EventType.SearchResultsAvailable</a></li><li>• "PartialSearchResultsAvailable" - <a href="#">PCCViewer.EventType.PartialSearchResultsAvailable</a></li></ul>
handler	function	The function that will be called whenever the event is triggered.

**Returns:**

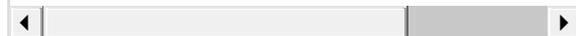
The SearchRequest object on which this method was called.

Type

[PCCViewer.SearchRequest](#)

**Example**

```
// subscribe  
searchRequest.on(PCCViewer.EventType.S  
onSearchCompleted);  
  
// handler declaration  
function onSearchCompleted(ev) {  
    alert("Search completed! Number  
of hits :" +  
searchRequest.getResults().length);  
}
```



## Class: SearchResult

### 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 <a href="#">PCCViewer.SearchRequest#getResults</a> to get results from a `SearchRequest` object.
	Use <a href="#">PCCViewer.SearchRequest#results</a> 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:	<a href="#">PCCViewer.SearchResult#getBoundingRectangle</a>
------	---

`(readonly) context :string`

Gets the search result text and some surrounding text.

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.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) `searchTerm`

:[PCCViewer.ViewerControl~SearchTerm](#) | [PCCViewer.ViewerControl~ProximitySearchTerm](#)

Gets the search term object from the `searchQuery` object that was passed to the [PCCViewer.SearchTask](#) constructor.

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.*

**Type:**

- number

See:

[PCCViewer.SearchResult#getStartIndexInContext](#)

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.

[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 =
```

```
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 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.

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:

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 Mon Mar 13 2017 16:48:45 GMT-0400 (Eastern Daylight Time)

## Class: SearchTask

### 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:

<code>searchQuery</code> string   <a href="#">PCCViewer.ViewerControl~SearchQuery</a>	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.

### 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 mark1
```

```
var results =
searchTask.search(mark1.mark1.getText(

//use it 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);
```

```
var results =  
searchTask.search(textString);
```

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:45 GMT-0400 (Eastern Daylight Time)

## Class: SearchTaskResult

### 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

(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`

`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
	<a href="#">PCCViewer.SearchTaskResult#getStartIndexInContext</a>
	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

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
```

```
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");
```

```
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 Mon Mar 13 2017 16:48:45 GMT-0400 (Eastern Daylight Time)

## Mixin: SessionData

Mixin: 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

- 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

```
KVPs 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 KVPs 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"><li>• This must be a string or undefined.</li><li>• The maximum length of the string is not limited by this function.</li></ul>

See:

[PCCViewer.SessionData#getSessionData](#)

[PCCViewer.SessionData#getSessionDataKeys](#)

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
```

## Class: SignatureControl

Class: SignatureControl

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 <b>must be visible</b> on the page when the SignatureControl 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
```

```
PCCViewer.SignatureControl(HTMLElement)  
  
// shorthand to use the query  
selector directly  
var signatureControl =  
PCCViewer.SignatureControl('#myDrawing
```

## 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

**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 Mon Mar 13 2017 16:48:45 GMT-0400 (Eastern Daylight Time)*

## Class: SignatureDisplay

Class: SignatureDisplay

`PCCViewer.SignatureDisplay`

{object}

Builds a DOM preview of the signature, to be used to display the signature outside of the viewer.

**Parameters:**

Name	Type	Description
domElement	HTMLElement	The element in which to insert signature preview.
signature	Object   <a href="#">PCCViewer.Signatures~FreehandSignature</a>   <a href="#">PCCViewer.Signatures~TextSignature</a>	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

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 Mon Mar 13 2017 16:48:45 GMT-0400 (Eastern Daylight Time)

## Class: ThumbnailControl

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

that `ViewerControl`.

## Constructor

```
new ThumbnailControl(domElement, viewerControl)
```

Creates a new `PCCViewer.ThumbnailControl` object.

### Parameters:

Name	Type	Description
<code>domElement</code>	<code>HTMLElement</code>	The DOM element in which to embed the <code>ThumbnailControl</code> .
<code>viewerControl</code>	<code>PCCViewer.ViewerControl</code>	The <code>ViewerControl</code> 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:

**PageSelectionChanged** string Event is triggered when the collection of selected thumbnails changes.

Augmented properties of the **PCCViewer.Event** object for this event:

- **pageNumbers** {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
eventType	string	A string specifying the event type. See <code>PCCViewer.ThumbnailControl.EventType</code> for a list and description of all supported events.
handler	<code>PCCViewer.Event~eventHandler</code>	A function that was attached previously to the <code>ViewerControl</code> .  <b>Note:</b> This must be the same function object previously passed to <code>PCCViewer.ThumbnailControl#on</code> . It cannot be a different object that is functionally equivalent.

See: `PCCViewer.ViewerControl#on`  
`PCCViewer.ViewerControl#off` for more

**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 <a href="#">PCCViewer.ThumbnailControl.EventType</a> for a list and description of all supported events.
handler	<a href="#">PCCViewer.Event~eventHandler</a>	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

*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:

pageNumber	number		The page to scroll to.								
options	Object	<optional>	Provide options to the scroll method to indicate desired behavior.								
<b>Properties</b>											
<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Attributes</th><th>Description</th></tr></thead><tbody><tr><td>forceAlignTop</td><td>boolean</td><td>&lt;optional&gt;</td><td>Overrides the default scrolling behavior, and forces the specified page to always be scrolled to the top of the view.</td></tr></tbody></table>				Name	Type	Attributes	Description	forceAlignTop	boolean	<optional>	Overrides the default scrolling behavior, and forces the specified page to always be scrolled to the top of the view.
Name	Type	Attributes	Description								
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:**

`pageNumbers` Array. 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 Mon Mar 13 2017 16:48:45 GMT-0400 (Eastern Daylight Time)

## Class: Viewer

Class: 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

---

```
$('#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 Mon Mar 13 2017 16:48:45 GMT-0400 (Eastern Daylight Time)

## Class: ViewerControl

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

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>	<code>PCCViewer.ViewerControl~ViewerControlOptions</code>	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

## **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

## 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

## (inner) TextSelection

A plain object convention describing a text selection.

### Properties:

Name	Type	Description
pageNumber	number	The page number that the selection starts on.
length	number	The length of the text.
text	string	The selected text. This is plain text without any formatting.
startIndex	number	The index at which the selection starts in the page.
rectangles	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	Type	Description
x	number	The x-coordinate of the top-left corner of the rectangle.
y	number	The y-coordinate of the top-left corner of the rectangle.
width	number	The width of the rectangle.
height	number	The height of the rectangle.
pageNumber	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
pageNumber	number	Indicates the page to which to add the mark.
markType	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
searchResult	PCCViewer.SearchResult	Indicates the search result that will include the location for the new mark.
markType	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(optionsopt) → {PCCViewer.BurnRequest}`

Burns redactions and signatures in the document.

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:

`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 which marks to burn regardless of their type.
burnSignatures	boolean	<optional>	true	Indicates whether signatures are required to be burned. <b>Note:</b> If <code>options.marks</code> exists, this parameter is ignored.
burnRedactions	boolean	<optional>	true	Indicates whether redactions are required to be burned. <b>Note:</b> If <code>options.marks</code> exists, this parameter is ignored.
burnAnnotations	boolean	<optional>	false	Indicates whether annotations are required to be burned. <b>Note:</b> If <code>options.marks</code> exists, this parameter is ignored.
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 burned document will be downloaded with a default filename.
removeFormFields	Array.<string>	<optional>		Specifies a list of the types of form fields to remove from the document. Currently, only a list including the value 'acroform' is

supported. If not set, then form fields are not removed.

See: [PCCViewer.BurnRequest](#) for more details on interacting with the burn process.

### Returns:

A result `BurnRequest` for this task.

Type

`PCCViewer.BurnRequest`

### Example

```
function onSuccessulBurn(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.

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:

Type

[PCCViewer.ViewerControl](#)

### Example

```
viewerControl.changeToPrevPage();
```

`clearMouseSelectedText(textSelection)` → [{PCCViewer.ViewerControl}](#)

Deselects the text provided in a TextSelected event.

### Parameters:

Name	Type	Description
textSelection	<a href="#">PCCViewer.ViewerControl~TextSelection</a>	The textSelection object provided in the TextSelected 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

**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}`

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   <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.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:  <b>Properties</b> <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

- `clientX` {number} The x window coordinate.
- `clientY` {number} The y window coordinate.

Type

Array.<Object> | Object

`convertToHighlight(searchResult) → {PCCViewer.Mark}`

Converts a search result into a highlight annotation.

**Parameters:**

Name	Type	Description
<code>searchResult</code>	<code>PCCViewer.SearchResult</code>	The search result to convert to a highlight.

**Throws:**

- If `PCCViewer.EventType.ViewerReady` event was not fired prior to using this method.

Type

Error

- If the provided parameter is not a valid `PCCViewer.SearchResult` object created in the current session.

Type

Error

**Returns:**

The new `PCCViewer.Mark`.

Type

`PCCViewer.Mark`

**Example**

```
var searchRequest = viewer.search("Hello");
var searchResult = searchRequest.getResults();
for (var i = 0; i < searchResult.length; i++) {
    var mark = viewer.convertToHighlight(searchResult[i]);
}
```

`convertToRedaction(searchResult) → {PCCViewer.Mark}`

Converts a search result into a text selection redaction.

**Parameters:**

Name	Type	Description
<code>searchResult</code>	<code>PCCViewer.SearchResult</code>	The search result to convert to a redaction.

**Throws:**

- If `PCCViewer.EventType.ViewerReady` event was not fired prior to using this method.

Type

Error

- If the provided parameter is not a valid `PCCViewer.SearchResult` object created in the

Type

Error

type  
Error

**Returns:**

The new `PCCViewer.Mark`.

Type

`PCCViewer.Mark`

**Example**

```
var searchRequest = viewer.search("Hello");
var searchResult = searchRequest.getResults();
for (var i = 0; i < searchResult.length; i++) {
    var mark = viewer.convertToRedaction(searchResult[i]);
}
```

`copyMarks(marks) → {Array.<PCCViewer.Mark>}`

Makes a copy of the specified marks.

**Parameters:**

Name	Type	Description
marks	<code>Array.&lt;PCCViewer.Mark&gt;</code>	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:**

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

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

### `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();
```

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
pageNumber	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 + '"');  
  }  
);
```

`enterTextMarkEditMode(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 null or undefined is given, then <b>all</b> 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,
```

```
// Put the mark in editing mode and scroll to the mark.  
viewerControl.enterTextMarkEditMode(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 =
```

`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.

## 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	<a href="#">PCCViewer.Mark</a>   <a href="#">PCCViewer.SearchResult</a>   Object	A mark or search result in a page, or an object containing pageNumber, x, and y 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

```
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;
```

**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	<code>PCCViewer.Mark.Type</code>   string	The mark type being requested. <b>Note:</b> 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.

**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:**

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

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;
```

```
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.

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}`

**Returns:**

The `PCCViewer.SearchResult` object. Returns null if no search result is selected.

Type

`PCCViewer.SearchResult` | null

`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.< <code>PCCViewer.Mark</code> >	An Array of objects of type <code>PCCViewer.Mark</code> .

**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

- [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
pageNumber	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")
}
```

`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, retainExistingMarksopt, markupLayeropt)` → `{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.

#### Parameters:

Name	Type	Attributes	Default	Description
<code>recordName</code>	string			Name of the annotation record to be loaded.
<code>retainExistingMarks</code>	boolean	<optional>	true	If true, retains the existing marks.
<code>markupLayer</code>	<code>PCCViewer.MarkupLayer</code>	<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
layerRecordIds	string   Array.<string>	A string or an array of layer record IDs.

### 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

**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
```

```
// 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	<code>PCCViewer.Mark</code>	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:**

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 <a href="#">PCCViewer.EventType</a> for a list and description of all supported events.
handler	<a href="#">PCCViewer.Event~eventHandler</a>	A function that was attached previously to the <a href="#">ViewerControl</a> .  <b>Note:</b> This must be the same function object previously passed to <a href="#">PCCViewer.ViewerControl#on</a> . It cannot be a 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 <a href="#">PCCViewer.EventType</a> 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.

### Parameters:

`options` Object <optional> Provides instructions for what to print. The object may have the following properties:

**Properties**

Name	Type	Attributes	Default	Description
range	string	<optional>	"all"	A string representing the pages to print. Pages are separated by commas, and ranges are separated by a hyphen. <ul style="list-style-type: none"> <li>• Sample value: "1, 2, 5-7, 9, 15-20"</li> <li>• Sample value: "all"</li> </ul>
orientation	string	<optional>	"portrait"	Describes the orientation of the printed pages. <ul style="list-style-type: none"> <li>• Possible values: "portrait" or "landscape"</li> </ul>
paperSize	string	<optional>	"letter"	Describes what size of paper this document should be printed on.
margins	string	<optional>	"default"	Whether to respect the default browser margins. This affects IE and Safari. <ul style="list-style-type: none"> <li>• Possible values:                             <ul style="list-style-type: none"> <li>◦ "default" When necessary, the pages will be smaller, so that the entire page content can fit on one printed page.</li> <li>◦ "none" Content will always be printed as an 8.5x11 inch page. The user is expected to set the browser print margins to 0.</li> </ul> </li> </ul>
includeMarks	boolean	<optional>	false	Whether to print marks. Includes both annotations and redactions. <ul style="list-style-type: none"> <li>• Possible values: true or false</li> <li>• Note: this value will be ignored if <code>PCCViewer.ViewerControl#canPrintMarks</code> returns false.</li> </ul>
includeAnnotations	boolean	<optional>	false	Whether to print annotations. <ul style="list-style-type: none"> <li>• Possible values: true or false</li> <li>• Note: this value will be ignored if <code>PCCViewer.ViewerControl#canPrintMarks</code> returns false.</li> </ul>
includeRedactions	boolean	<optional>	false	Whether to print redactions. <ul style="list-style-type: none"> <li>• Possible values: true or false</li> <li>• Note: this value will be ignored if <code>PCCViewer.ViewerControl#canPrintMarks</code> returns false.</li> </ul>
includeComments	string	<optional>	"none"	Location to print comments. <ul style="list-style-type: none"> <li>• Possible values: "none", "followingPage", or "documentEnd"</li> </ul>
includeReasons	string	<optional>	"none"	Location to print redaction reasons. <ul style="list-style-type: none"> <li>• Possible values: "none", "followingPage", or "documentEnd"</li> </ul>
redactionViewMode	string	<optional>	"Normal"	Whether to print document content text underneath solid rectangle redactions and selection text redactions marks. <ul style="list-style-type: none"> <li>• Possible values:                             <ul style="list-style-type: none"> <li>◦ "Normal" The document context text underneath redaction marks is not printed. The redaction marks are opaque.</li> <li>◦ "Draft" Document content text underneath rectangle redactions</li> </ul> </li> </ul>

and text selection redactions will be visible in the printed document.

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(conversationsopt)` → `{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	<code>PCCViewer.Conversation</code>   Array. < <code>PCCViewer.Conversation</code> >	<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 false value.  
Type

**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>	This optional parameter specifies conversion options to be used for converting the format of the document.

**Properties**

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.ConversionCompleted,

    function(ev) {
        alert("Document conversion completed.");
    })

.on(PCCViewer.ConversionRequest.EventType.ConversionProgress,
```

```
"%");
    })

    .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
pageNumber	number	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 `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].name);
    }
    alert(namesArray.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);
  }
);
```

```
        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>	number	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));
    }
);
```

`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.

Type

`Error`

- If the `PCCViewer.EventType.ViewerReady` event was not fired prior to using this method.

Type

`Error`

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

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>	<code>string</code>		Name of the annotation record. The markup is saved to an XML file using a filename that includes a unique ID for the document being viewed and the provided annotation record name, so valid filename characters are required. When later viewing the document, the saved markup can be loaded by passing the record name to the <code>PCCViewer.ViewerControl#loadMarkup</code> method.
<code>options</code>	<code>PCCViewer.ViewerControl~SaveMarkupOptions</code>	<optional>	This optional parameter specifies options for the markup types to be saved. The <code>PCCViewer.ViewerControl~SaveMarkupOptions</code> details the options object.

#### 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}`

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
id	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**

```
viewerControl.scrollTo(target);
```

`scrollTo(target)` → `{PCCViewer.ViewerControl}`

Scrolls to the specified object.

**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 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.

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 pageNumber, x, and y 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   <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.search("Hello");

// Subscribe to the PartialSearchResultsAvailable event to
get search results as they become available.
searchRequest.on('PartialSearchResultsAvailable',
function(_event) {
```

```
});  
  
// 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.enterTextMarkEditMode(textAnnotation).selectf
```

`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

**Parameters:**

Name	Type	Description
marks	Array. <PCCViewer.Mark>	An array of <a href="#">PCCViewer.Mark</a> 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	<a href="#">PCCViewer.Mark</a>   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 them
var marksToUse = viewerControl.getSelectedMarks();

viewerControl.serializeMarks(marksToUse).then(
```

```
    // 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);
```

```
function(_event) {  
    var newResults = [];  
    //Retrieve results  
    newResults = _event.partialSearchResults;  
});
```

## setActiveMarkupLayer(A)

Sets the viewer control's active markup layer.

### Parameters:

Name	Type	Description
A	<a href="#">PCCViewer.MarkupLayer</a>	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:

**factoryFunction** function 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:

- 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

If this function returns a false 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 false, then the viewer control will throw an exception when attempting to create a conversation DOM element.

This function has three parameters, in this order:

- `conversation` (`PCCViewer.Conversation`) - The Conversation in which to generate a DOM element.
- `state` (Object) - An object that indicates the state of the conversation. This object has an `isSelected` boolean property.
- `existingDOM` (HTMLElement | undefined) - 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 `undefined` 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';
```

```
// 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].get
// 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(conversationFactoryFur
```

`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](#)

`setMarkHandleMode(markHandleMode) → {PCCViewer.ViewerControl}`

Sets the mark handle mode.

#### Parameters:

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.HideC  
setPageLayout(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	<a href="#">PCCViewer.PageLayout</a>	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:

**pageNumber** number | The 1-based page number of the page. A string representation of a valid string 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
<code>redactionViewMode</code>	<code>PCCViewer.RedactionViewMode</code>	Displays redaction marks showing/hiding underneath document content text.

**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.RedactionViewMode
```

```
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
scaleFactor	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.

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.getConversation());
```

`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
searchResult	<a href="#">PCCViewer.SearchResult</a>   null			The search result object from the results object. If a value of null is passed, then no search results will be selected.
scrollTo	boolean	<optional>	false	If true, the viewer will navigate to the page with the search result. If the value of searchResult is null, 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

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);
});
```

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	<a href="#">PCCViewer.ViewMode</a>	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

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</code>   <code>PCCViewer.ViewerControl~SearchQuery</code>	A string or a <code>SearchQuery</code> object that requires validation. See <code>PCCViewer.ViewerControl#search</code> .

### 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,
        wildcard: 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](#)

- if the zoomFactor value is invalid.

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

```
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	Type	Attributes	Default	Description
type	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.
distance	number			Number of intermediate words between the search terms.
terms	Array. < <code>PCCViewer.ViewerControl~SearchTerm</code> >			An array of search terms.
contextPadding	number	<optional>	25	The maximum number of leading and trailing character in the context string ( <code>PCCViewer.SearchResult#getContext</code> ) that is given for a search result.
highlightColor	string	<optional>		The highlight color of the search results matching this search term ( <code>PCCViewer.SearchResult#getHighlightColor</code> ). If not defined, then a random color will be 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 = {
```

```
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	Type	Attributes	Default	Description
includeAnnotations	boolean	<optional>	true	Indicates whether annotation mark types should be saved.
includeSignatures	boolean	<optional>	false	Indicates whether signature mark types should be saved.
includeRedactions	boolean	<optional>	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	Type	Attributes	Default	Description
endsWith	boolean	<optional>	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>	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 true, then the following options are ignored:

			<ul style="list-style-type: none"> <li>• <code>endsWith</code></li> </ul>
<code>exactPhrase</code>	<code>boolean &lt;optional&gt;</code>	<code>true</code>	<p>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.</p> <p>If this value is <code>true</code>, then the following options are ignored:</p> <ul style="list-style-type: none"> <li>• <code>wildcard</code></li> </ul>
<code>matchCase</code>	<code>boolean &lt;optional&gt;</code>	<code>false</code>	Indicates whether matching is case sensitive.
<code>matchWholeword</code>	<code>boolean &lt;optional&gt;</code>	<code>false</code>	<p>Match a phrase that starts and ends on word boundaries.</p> <p>If this value is <code>true</code>, then the following options are ignored:</p> <ul style="list-style-type: none"> <li>• <code>wildcard</code></li> <li>• <code>endsWith</code></li> <li>• <code>beginsWith</code></li> </ul>
<code>wildcard</code>	<code>boolean &lt;optional&gt;</code>	<code>false</code>	<p>A value that indicates whether the search term includes wild cards.</p> <p>Supported wildcard characters are:</p> <ul style="list-style-type: none"> <li>• <code>*</code> - match one or more characters</li> <li>• <code>?</code> - match one character</li> </ul> <p>If this value is <code>true</code>, then the following options are ignored:</p> <ul style="list-style-type: none"> <li>• <code>endsWith</code></li> <li>• <code>beginsWith</code></li> </ul>

### 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	Type	Description
<code>searchTerms</code>	Array. <(PCCViewer.ViewerControl~SearchTerm PCCViewer.ViewerControl~ProximitySearchTerm)>	An array of search terms.

```
// 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	Type	Attributes	Default	Description
searchTerm	string			A string value to match against. This may be treated as a string literal, a string with wild cards, or a regular expression. See <a href="#">searchTermIsRegex</a> and <a href="#">PCCViewer.ViewerControl~SearchMatchingOptions</a> .
contextPadding	number	<optional>	25	The maximum number of leading and trailing character in the context string ( <a href="#">PCCViewer.SearchResult#getContext</a> ) that is given for a search result.
highlightColor	string	<optional>		The highlight color of the search results matching this search term ( <a href="#">PCCViewer.SearchResult#getHighlightColor</a> ).  If not defined, then a color will be chosen. A different color will be chosen for each search term, and if <code>matchingOptions.exactPhrase === false</code> , then a different color will be chosen for each word in the search term.
matchingOptions	<a href="#">PCCViewer.ViewerControl~SearchMatchingOptions</a>	<optional>		Options that specify how the search term will be matched.
searchTermIsRegex	boolean	<optional>	false	Indicates if the search term is treated as a regular expression.  If this value is true, then the following matching options can be used: <ul style="list-style-type: none"> <li>• <code>matchingOptions.matchCase</code></li> </ul> Note that it is invalid to set the other matching options for a regular expression search. Doing so may make search behave in unexpected ways.

### 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.
```

```
};
```

## 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	Type	Attributes	Default	Description															
documentID	string			REQUIRED The ID of the document to load.															
imageHandlerUrl	string			REQUIRED The end point of the web tier services that support the viewer.															
language	Object	<optional>		Specifies the language to use for the text in the ViewerControl.  The options here are a subset of the <a href="#">external:jQuery.fn~LanguageOptions</a> object. These values are used directly by the ViewerControl, and are passed in by the jQuery Plugin, <a href="#">external:jQuery.fn#pccViewer</a> . When creating a custom UI using ViewerControl directly, these options will need to be passed in during initialization. If they language element is not present with the following properties, ViewerControl will use the English language defaults.  <b>Properties</b> <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>pageLoadFailed</td> <td>string</td> <td>&lt;optional&gt;</td> <td>"Page Load Failed"</td> <td>Text to use for "Page Load Failed"</td> </tr> <tr> <td>pageLoadFailedRetry</td> <td>string</td> <td>&lt;optional&gt;</td> <td>"Retry"</td> <td>Text to use for "Retry"</td> </tr> </tbody> </table>	Name	Type	Attributes	Default	Description	pageLoadFailed	string	<optional>	"Page Load Failed"	Text to use for "Page Load Failed"	pageLoadFailedRetry	string	<optional>	"Retry"	Text to use for "Retry"
Name	Type	Attributes	Default	Description															
pageLoadFailed	string	<optional>	"Page Load Failed"	Text to use for "Page Load Failed"															
pageLoadFailedRetry	string	<optional>	"Retry"	Text to use for "Retry"															
viewMode	string	<optional>	"Document"	The mode used to view documents containing different sized pages. See the <a href="#">PCCViewer.ViewMode</a> enumeration for details on each view mode.															
pageLayout	string	<optional>	"Vertical"	The layout used to arrange pages in the viewer. See the <a href="#">PCCViewer.PageLayout</a> enumeration for details on each page layout.															
pageRotation	number	<optional>	0	The amount in degrees clockwise to rotate each page. Valid values are multiples of 90: ..., -270, -180, -90, 0, 90, 180, 270, ...															
resourcePath	string	<optional>	"img"	The location of the images within the viewer. This is the folder that holds the ArtMarkHandles.png and EditTextMark.png files. This path should be relative to the page that the viewer is embedded on.															
printTemplate	string	<optional>	""	A text representation of a full web page used for printing purposes. It is recommended that this value be set to the content of the file "printTemplate.html". If this value is set to an empty string (default), then printing will be unavailable.															
template.print	string	<optional>	""	This is an alias for printTemplate. If printTemplate is not available in the options object, the control will look for this property instead. This property matches the value as it is used in the jQuery plugin options ( <a href="#">external:jQuery.fn~Options</a> ).															
encryption	boolean	<optional>	false	Specifies whether the viewer should use encryption. See the help topic "Enabling Content Encryption" for more details.															
serviceResponseTimeout	number	<optional>	60000	Indicates the response timeout interval for all services to the server. A value of zero indicates the default browser value will be used.															
debug	boolean	<optional>	false	Indicates whether the viewer should log its actions.															
RedactionViewMode	string	<optional>	"Normal"	The redaction view mode can be used to view document content text underneath the opaque rectangle redaction marks and the text selection redaction marks. See the <a href="#">PCCViewer.RedactionViewMode</a> enumeration for details on redaction view modes.															
markHandleMode	string	<optional>	"HideSideHandlesWhenClose"	The mark handle mode. See the <a href="#">PCCViewer.MarkHandleMode</a> enumeration for details on each mark handle mode.															

<code>discardPageText</code>	boolean	<optional>	false	Specifies whether text on pages that are not displayed is discarded from memory. Note that it is necessary to use the <code>PCCViewer.ViewerControl#requestPageText</code> method to request a text page before using any API that requires the text of a page that is not currently displayed.
<code>searchMethodType</code>	string	<optional>	"auto"	The type of search that will be used by the <code>PCCViewer.ViewerControl#search</code> method. Note that client search will be used if server search is unavailable (this is the case when using viewing packages). Possible values are: <ul style="list-style-type: none"><li>"server" - Use the <code>PCCViewer.ViewerControl#serverSearch</code> API internally.</li><li>"client" - Use the <code>PCCViewer.ViewerControl#clientSearch</code> API internally.</li><li>"auto" - Viewer Control will use <code>PCCViewer.ViewerControl#serverSearch</code> when the page count is above the value set in the <code>searchMethodPageCountThreshold</code> option. Viewer Control will use <code>PCCViewer.ViewerControl#clientSearch</code> otherwise.</li></ul>
<code>searchMethodPageCountThreshold</code>	number	<optional>	80	The maximum number of pages a document can have for client-side search. This value is used when <code>searchMethodType</code> is set to "auto". Must be greater than 1.
<code>viewerAssetsPath</code>	string	<optional>	"viewer-assets"	The path to the viewer-assets folder, which is used in the print template. The text <code>{{viewerAssetsPath}}</code> in your print template will be replaced by the value specified for this option.

Documentation generated by *JSDoc 3.3.3* on Wed Mar 29 2017 10:18:08 GMT-0400 (Eastern Daylight Time)

## Mixin: Data

### 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

**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
```

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  
KVPs 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 KVPs 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:

key string The key for which to set the data value.

value string This is the value to set for the key.

- 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
```

```
throws  
item.setData("FooBar", true); //  
throws  
item.setData("FooBar", {}); //  
throws  
item.setData("FooBar", []); //  
throws
```

Documentation generated by [JSDoc 3.3.3](#) on Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Namespace: Ajax

### 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.

3. 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.

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.
- 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.

### Parameters:

Name	Type	Description
headers	Object	An object containing the headers we will send with every request. <ul style="list-style-type: none"><li>• This must be an object or undefined.</li></ul>

See: [PCCViewer.Ajax#getHeaders](#)

### Throws:

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:**

method `PCCViewer.Ajax~OverrideMethod` A function that will be called with every request that allows you to override or modify the request.

- 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('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	<b>Properties</b> <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>&lt;optional&gt;</td><td>"GET"</td><td></td></tr><tr><td>headers</td><td>Object</td><td>&lt;optional&gt;</td><td>{}</td><td></td></tr><tr><td>body</td><td>String</td><td>&lt;optional&gt;</td><td>null</td><td></td></tr><tr><td>timeout</td><td>Number</td><td>&lt;optional&gt;</td><td>null</td><td></td></tr><tr><td>mimeType</td><td>String</td><td>&lt;optional&gt;</td><td>null</td><td><ul style="list-style-type: none"><li>The mime type used for the response, overriding what the server sends. See: <a href="https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest#overrideMimeType">https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest#overrideMimeType</a></li></ul></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"><li>The mime type used for the response, overriding what the server sends. See: <a href="https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest#overrideMimeType">https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest#overrideMimeType</a></li></ul>
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"><li>The mime type used for the response, overriding what the server sends. See: <a href="https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest#overrideMimeType">https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest#overrideMimeType</a></li></ul>																																	

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(option  
{  
  options.url = 'http://accusoft.com/';  
});  
  
// This method can prevent default execution  
from continuing  
PCCViewer.Ajax.setOverrideMethod(function(option  
{  
  var deferred = $.Deferred();
```

```
        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 Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Namespace: 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.

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	<a href="#">externaljQuery.fn~Options</a>	<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
Minute	hh	01 02 ... 11 12
	m	0 1 ... 58 59
AM/PM	mm	00 01 ... 58 59
	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, `externaljQuery.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	Type	Attributes	Default	Description
documentID	string			The ID of the document to load. This option is a part of the <code>PCCViewer.ViewerControl~ViewerControlOptions</code> options object.

# PrizmDoc v12.3 - Updated June 23, 2017

956

			redacted document. See <a href="#">PCCViewer.ViewerControl#burnMarkup</a> for more information. Example: "sample.doc"
imageHandlerUrl	string	<optional>	"/pcc.ashx"
language	<a href="#">externaljQuery.fn~LanguageOptions</a>		
redactionReasons	<a href="#">externaljQuery.fn~redactionReasons</a>	<optional>	
annotationsMode	string	<optional>	"LegacyAnnotations"
annotationID	string	<optional>	
autoLoadAnnotation	boolean	<optional>	false
autoLoadAllLayers	boolean	<optional>	false
editableMarkupLayerSource	string	<optional>	
editableMarkupLayerValue	string	<optional>	
lockEditableMarkupLayer	boolean	<optional>	false
template	Object		

The end point of the web tier services that support the viewer. Unless specified, viewer.js will assume it is running in the default .NET sample. This option is a part of the [PCCViewer.ViewerControl~ViewerControlOptions](#) options object.

Specifies the language to use for the text in the ViewerControl. Use this option to localize the viewer.

A list of reasons to be used for the redaction reasons controls.

The annotationsMode specifies the mode in which the annotations will be handled in the viewer. When the annotationsMode is set to "LegacyAnnotations" (default), all annotations will be handled as in the releases prior to 10.3. The second option is "LayeredAnnotations". This option supports the new layered annotations feature available in the releases 10.3 and higher. A convenience enumeration for these two annotation mode types [annotationsModeEnum](#) is available in the viewer.js file and as a property of the Viewer object. This enumeration is not available in the API.

**Deprecation Notice:** In the future, the "LegacyAnnotations" option will be deprecated.

Specifies the annotation file to be used within the viewer.  
*This property is only observed when annotationsMode is set to "LegacyAnnotations".*

If set to true, the specified annotation file will be loaded when the viewer launches.  
*This property is only observed when annotationsMode is set to "LegacyAnnotations".*

When set to true, all available layers will be loaded into the document. This includes all layers returned by [requestMarkupLayerNames](#) and [getSavedMarkupNames](#).  
*This property is only observed when annotationsMode is set to "LayeredAnnotations".*

When set to "LayerRecordId", the layer with the record ID specified by the [editableMarkupLayerValue](#) will be loaded from JSON into the document. When set to "XmlName", the layer with the name specified by the [editableMarkupLayerValue](#) viewer parameter will be loaded from XML into the document, unless the XML layer with that name has been saved to JSON, in which case the layer will be loaded from JSON. When set to "DefaultName", a new empty layer is loaded into the document but given the name specified by the [editableMarkupLayerValue](#) viewer parameter.  
*This property is only observed when annotationsMode is set to "LayeredAnnotations" and editableMarkupLayerValue is set to a valid value.*

When the [editableMarkupLayerSource](#) viewer parameter is set to "LayerRecordId", this specifies the record ID of the layer that will be loaded from JSON into the document. When the [editableMarkupLayerSource](#) viewer parameter is set to "XmlName", this specifies the name of the XML layer that will be loaded from XML into the document, unless the XML layer with that name has been saved to JSON, in which case the layer will be loaded from JSON. When the [editableMarkupLayerSource](#) viewer parameter is set to "DefaultName", this specifies name to give the new empty layer.  
*This property is only observed when annotationsMode is set to "LayeredAnnotations" and editableMarkupLayerSource is set to a valid value.*

When set to true, the buttons that open the load annotations (for edit) dialog will be removed so that it is not possible to change the editable markup layer. The menu item that opens the edit layer name dialog will also be removed so it is not possible to rename the editable markup layer.

This objects holds the various templates that the viewer UI needs. The templates correspond to HTML files available in all samples -- the filenames use the same name as the template names here, appending "Template.html" to the end -- for example, the template `fileName` would correspond to a `fileNameTemplate.html` file.

**Properties**

Name	Type	Description
viewer	string	This is the main viewer template. It contains partial HTML, and is parsed using Underscore to complete template variables.
contextMenu	string	This is the template for the floating menu used to edit annotations. It contains partial HTML, and is parsed using Underscore to complete template variables.
overwriteOverlay	string	This is the template used for the overlay menu when the Viewer detects that the user is saving markup using a name that already exists. It contains partial HTML, and is parsed using Underscore to complete template variables.
unsavedChangesOverlay	string	This is the template used for the overlay menu when the Viewer detects that the user is opening a markup file while having unsaved markup already loaded in the viewer. It contains partial HTML, and is parsed using Underscore to complete template variables.
printOverlay	string	This is the template used for the overlay menu displayed when the user selects to print the document. It contains partial HTML, and is parsed using Underscore to complete template variables.
print	string	This is the template used to enable the viewer to print. This template contains a full HTML page. Variables are parsed using curly brackets. This template is part of the main <a href="#">ViewerControlOptions</a> , <a href="#">PCCViewer.ViewerControl~ViewerControlOptions</a> .
downloadOverlay	string	This is the template used for the overlay menu displayed when the user selects to download the document. It contains partial HTML, and is parsed using Underscore to complete template variables.
esignOverlay	string	This is the template used for the overlay menu displayed when the user selects to manage e-signatures. It contains partial HTML, and is parsed using Underscore to complete template variables.
comment	string	This is the template for comments displayed in the ViewerControl comments panel. It contains partial HTML, and is parsed using Underscore to complete template variables.
copyOverlay	string	This is the template used for the clipboard overlay when a user attempts to copy document text using a touch device. It contains

			template variables.																																																												
			<p>hyperlinkMenu string This is the template used to render the menu that appears when a user creates or views a hyperlink annotation. It contains partial HTML, and is parsed using Underscore to complete template variables.</p> <p>imageStampOverlay string This is the template used for the image stamp picker for the image stamp annotations and redactions. It contains partial HTML, and is parsed using Underscore to complete template variables.</p> <p>pageRedactionOverlay string This is the template used for the full page redaction selection dialog. It contains partial HTML, and is parsed using Underscore to complete template variables.</p>																																																												
uiElements	Object	<optional>	<p>This object contains directives to easily hide or disable default UI elements in the viewer. This option is an alternative to removing the elements entirely from the <code>viewerTemplate.html</code> file, and enables having features that are conditionally available.</p> <p>When the viewer removes various elements, it looks for a <code>data-pcc-removable-id</code> to be defined on the HTML element. This id will match to the key defined in the <code>uiElements</code> object. Setting the value of that key to <code>true</code>, or not including the key at all, results in the respective elements being visible and active in the viewer. Setting the key to <code>false</code> will cause the viewer to remove that element at runtime.</p> <p>This feature is enabled for <code>viewerTemplate.html</code> and <code>contextMenuTemplate.html</code>. The following <code>data-pcc-removable-id</code> keys are already defined by default:</p> <p><b>Properties</b></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>annotateTab</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the Annotate Tab.</td> </tr> <tr> <td>redactTab</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the Redact Tab.</td> </tr> <tr> <td>searchTab</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the Search Tab.</td> </tr> <tr> <td>viewTab</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the View Tab.</td> </tr> <tr> <td>esignTab</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the E-Sign Tab.</td> </tr> <tr> <td>copyPaste</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the Text Select Tool.</td> </tr> <tr> <td>download</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the Download Button.</td> </tr> <tr> <td>printing</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the Print Button.</td> </tr> <tr> <td>advancedSearch</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>false</td> <td>Enables the advanced search features, including searching through marks and comments.</td> </tr> <tr> <td>attachments</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>If true, the viewer will automatically load attachments of the currently loaded document.</td> </tr> <tr> <td>fullScreenOnInit</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>false</td> <td>Specifies whether the viewer will fill the browser window when initialized. If this is not defined or set to false the viewer will use the width and height set in <code>viewer.css</code>.</td> </tr> </tbody> </table>	Name	Type	Attributes	Default	Description	annotateTab	boolean	<optional>	true	Show or hide the Annotate Tab.	redactTab	boolean	<optional>	true	Show or hide the Redact Tab.	searchTab	boolean	<optional>	true	Show or hide the Search Tab.	viewTab	boolean	<optional>	true	Show or hide the View Tab.	esignTab	boolean	<optional>	true	Show or hide the E-Sign Tab.	copyPaste	boolean	<optional>	true	Show or hide the Text Select Tool.	download	boolean	<optional>	true	Show or hide the Download Button.	printing	boolean	<optional>	true	Show or hide the Print Button.	advancedSearch	boolean	<optional>	false	Enables the advanced search features, including searching through marks and comments.	attachments	boolean	<optional>	true	If true, the viewer will automatically load attachments of the currently loaded document.	fullScreenOnInit	boolean	<optional>	false	Specifies whether the viewer will fill the browser window when initialized. If this is not defined or set to false the viewer will use the width and height set in <code>viewer.css</code> .
Name	Type	Attributes	Default	Description																																																											
annotateTab	boolean	<optional>	true	Show or hide the Annotate Tab.																																																											
redactTab	boolean	<optional>	true	Show or hide the Redact Tab.																																																											
searchTab	boolean	<optional>	true	Show or hide the Search Tab.																																																											
viewTab	boolean	<optional>	true	Show or hide the View Tab.																																																											
esignTab	boolean	<optional>	true	Show or hide the E-Sign Tab.																																																											
copyPaste	boolean	<optional>	true	Show or hide the Text Select Tool.																																																											
download	boolean	<optional>	true	Show or hide the Download Button.																																																											
printing	boolean	<optional>	true	Show or hide the Print Button.																																																											
advancedSearch	boolean	<optional>	false	Enables the advanced search features, including searching through marks and comments.																																																											
attachments	boolean	<optional>	true	If true, the viewer will automatically load attachments of the currently loaded document.																																																											
fullScreenOnInit	boolean	<optional>	false	Specifies whether the viewer will fill the browser window when initialized. If this is not defined or set to false the viewer will use the width and height set in <code>viewer.css</code> .																																																											
immediateActionMenuMode	string	<optional> "off"	<p>Sets the mode of the immediate action menu. The following options are available:</p> <ul style="list-style-type: none"> <li>"on": The menu will appear after creating a mark or selecting text, close to the mouse cursor, allowing the user to take quick actions.</li> <li>"hover": In supported browsers, an icon will appear after creating a mark or selecting text. When hovering over this icon, the full menu will appear. On mobile viewports and in IE8, this will be the same as "on".</li> <li>"off": The menu will not appear.</li> </ul>																																																												
immediateActionMenuActionsFilter	object	<optional>	<p>When <code>immediateActionMenuMode</code> is set to "on" or "hover", this list will be used to define which actions are available in the menu.</p> <p><b>Properties</b></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>comment</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the Add Comment Button.</td> </tr> <tr> <td>select</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the Select Button.</td> </tr> <tr> <td>copy</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the Copy Button.</td> </tr> <tr> <td>highlight</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the Highlight Button.</td> </tr> <tr> <td>redact</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the Redact Button.</td> </tr> <tr> <td>hyperlink</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the Hyperlink Button.</td> </tr> <tr> <td>cancel</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>true</td> <td>Show or hide the Cancel Button.</td> </tr> </tbody> </table>	Name	Type	Attributes	Default	Description	comment	boolean	<optional>	true	Show or hide the Add Comment Button.	select	boolean	<optional>	true	Show or hide the Select Button.	copy	boolean	<optional>	true	Show or hide the Copy Button.	highlight	boolean	<optional>	true	Show or hide the Highlight Button.	redact	boolean	<optional>	true	Show or hide the Redact Button.	hyperlink	boolean	<optional>	true	Show or hide the Hyperlink Button.	cancel	boolean	<optional>	true	Show or hide the Cancel Button.																				
Name	Type	Attributes	Default	Description																																																											
comment	boolean	<optional>	true	Show or hide the Add Comment Button.																																																											
select	boolean	<optional>	true	Show or hide the Select Button.																																																											
copy	boolean	<optional>	true	Show or hide the Copy Button.																																																											
highlight	boolean	<optional>	true	Show or hide the Highlight Button.																																																											
redact	boolean	<optional>	true	Show or hide the Redact Button.																																																											
hyperlink	boolean	<optional>	true	Show or hide the Hyperlink Button.																																																											
cancel	boolean	<optional>	true	Show or hide the Cancel Button.																																																											
commentsPanelMode	string	<optional> "auto"	<p>Sets the mode of the comments panel. The following options are available:</p> <ul style="list-style-type: none"> <li>"full": The entire content of the comments are displayed in the sidebar of the document.</li> <li>"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.</li> <li>"auto": This mode will intelligently switch between the full and skinny mode, in order to optimize the space available for viewing the document.</li> </ul>																																																												
stickyTools	Object	<optional> "default"	<p>Sets the mode for the sticky tools behavior.</p> <p>The following options are available:</p> <ul style="list-style-type: none"> <li>"on": Sticky tools will always be on. The tool will remain active until the user switches it.</li> <li>"off": Sticky tools will always be off. The tool will remain active for one use, where relevant.</li> <li>"default": Clicking on a mouse tool button will toggle between sticky and non-sticky mode. The first click will trigger the tool in non-sticky mode, and every following click will toggle sticky mode on and off.</li> </ul> <p>See the <code>stickyToolsFilter</code> option for more information.</p>																																																												

			<p>sticky behavior. By default, all drawing tools (such as EllipseAnnotation, RectangleRedaction, etc.) will have this enhanced behavior enabled. Tools that do not appear on this list (and are not part of the defaults), or tools set to false, will never exhibit the sticky behavior.</p> <p>This object can have any of the values listed in <code>PCCViewer.MouseTool.Type</code>, using the same keys as that enumerable. The values for these keys are true for enabled, and false for disabled.</p> <p>See the <code>stickyTools</code> option for more information.</p> <p><i>Note that for some tools, like Magnifier or PanAndEdit, this mode is not relevant, as these tools always remain active until the user chooses a new tool. Keeping them on or off the list will not change that tool's behavior.</i></p> <p>Example: { RectangleAnnotation: true, PlaceSignature: false }</p>																														
signatureCategories	string	<optional>	<p>Specifies the categories of eSignature to add to the UI. This is a comma separated string containing the types of signatures that the application expects. When no value is passed in, the UI to select categories will be disabled.</p> <p>Example: "Full Signature, Initials, Name, Title"</p>																														
commentDateFormat	<a href="#">externaljQuery.fn~DateFormat</a>	<optional>	<p>Specifies the date format to use to display comment dates.</p> <p>"MM/DD/YYYY h:mma"</p>																														
signatureDateFormat	<a href="#">externaljQuery.fn~DateFormat</a>	<optional>	<p>Specifies the date format to use to display date signatures.</p> <p>"MM/DD/YYYY"</p>																														
predefinedSearch	Object	<optional>	<p>Specifies options so that the viewer can prepopulate the document with search terms. An example of this is defined in the predefinedSearch.json file available in all HTML5 viewer samples.</p> <p><b>Properties</b></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>highlightColor</td> <td>string</td> <td>&lt;optional&gt;</td> <td></td> <td>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"</td> </tr> <tr> <td>searchOnInit</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td></td> <td>Whether to run the search when the viewer is initialized. Only search terms that use <code>selected: true</code> will be searched for when this property is enabled.</td> </tr> <tr> <td>fixed</td> <td>boolean</td> <td>&lt;optional&gt;</td> <td>false</td> <td>The default fixed value of the search terms. This is overridden by the term-level parameter. If set to true, the search terms are always included when performing a search.</td> </tr> <tr> <td>globalOptions</td> <td>Object</td> <td>&lt;optional&gt;</td> <td></td> <td></td> </tr> <tr> <td>terms</td> <td>Array. <a href="#">externaljQuery.fn~predefinedSearchTerm</a></td> <td>&lt;optional&gt;</td> <td></td> <td>An array of search terms to populate predefined search.</td> </tr> </tbody> </table>	Name	Type	Attributes	Default	Description	highlightColor	string	<optional>		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"	searchOnInit	boolean	<optional>		Whether to run the search when the viewer is initialized. Only search terms that use <code>selected: true</code> will be searched for when this property is enabled.	fixed	boolean	<optional>	false	The default fixed value of the search terms. This is overridden by the term-level parameter. If set to true, the search terms are always included when performing a search.	globalOptions	Object	<optional>			terms	Array. <a href="#">externaljQuery.fn~predefinedSearchTerm</a>	<optional>		An array of search terms to populate predefined search.
Name	Type	Attributes	Default	Description																													
highlightColor	string	<optional>		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"																													
searchOnInit	boolean	<optional>		Whether to run the search when the viewer is initialized. Only search terms that use <code>selected: true</code> will be searched for when this property is enabled.																													
fixed	boolean	<optional>	false	The default fixed value of the search terms. This is overridden by the term-level parameter. If set to true, the search terms are always included when performing a search.																													
globalOptions	Object	<optional>																															
terms	Array. <a href="#">externaljQuery.fn~predefinedSearchTerm</a>	<optional>		An array of search terms to populate predefined search.																													
searchResultsPageLength	number	<optional>	<p>The number of search results to show at a time in the search results panel. The user can use the search results navigation buttons to page through the search results. If a positive number is not specified, the default of 250 is used.</p>																														

predefinedSearchTerm

**Type:**

- Object

**Properties:**

Name	Type	Attributes	Default	Description
searchTerm	string			The term to use for search. If used along with userDefinedRegex, this becomes the name of the Regular Expression, while the Regular Expression is used to perform the search.
highlightColor	string	<optional>		The color to use when highlighting results from this particular search.
fixed	boolean	<optional>	false	Whether or not the search term is always included when performing a search. If set to true, the value of predefinedSearchTerm.selected is disregarded since the search term is always included.
options	Object	<optional>		This is the same options object, overriding the settings in globalOptions for this specific term.
userDefinedRegex	string	<optional>		A regular expression that will be searched in place of searchTerm. The first and last forward slashes, as well as the flags, are stripped from the string. For example, "/Pa(w+)/ig" will become "Pa(w+)".  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 "Pa(w+)", this regular expression should be properly escaped like this "Pa\\(w+)".  All patterns use the Global (g) flag.
selected	boolean	<optional>	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 searchOnInit: true.

**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	Type	Attributes	Default	Description															
enableRedactionReasonSelection	boolean	<optional>	true	When set to false, the reason selection UI will not be enabled.															
reasons	Array. <Object>			An array of redaction reason objects. Each object will have the following properties:  <b>Properties</b> <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>reason</td><td>string</td><td></td><td></td><td>The reason.</td></tr><tr><td>defaultReason</td><td>boolean</td><td>&lt;optional&gt;</td><td>false</td><td>Specifies that this is the default reason. Only one item in the array can be set to true.</td></tr></tbody></table>	Name	Type	Attributes	Default	Description	reason	string			The reason.	defaultReason	boolean	<optional>	false	Specifies that this is the default reason. Only one item in the array can be set to true.
Name	Type	Attributes	Default	Description															
reason	string			The reason.															
defaultReason	boolean	<optional>	false	Specifies that this is the default reason. Only one item in the array can be set to true.															
autoApplyDefaultReason	boolean	<optional>	false	When set to true, the viewer will automatically apply the default reason to redactions.															
enableFreeformRedactionReasons	boolean	<optional>	false	When set to true, users may type a redaction reason instead of selecting one from a list.															
maxLengthFreeformRedactionReasons	number	<optional>	false	Sets the maximum length of a typed redaction reason.															

Documentation generated by JSDoc 3.3.3 on Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Namespace: Language

### Namespace: Language

#### PCCViewer. Language

A global object that defines the language data used by the PrizmDoc Viewer.

### 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("annotateL
```

### 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("annotatel

// Look up language data using a key
with dot notation.
myValue =
PCCViewer.Language.getValue("printDial
```

**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"
// ...
```

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:45 GMT-0400 (Eastern Daylight Time)

## Namespace: MarkSchema

### 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	Type	Description
rectangle	<b>MarkSchema~Rectangle</b>	The position of the mark on the page.
pageData	<b>MarkSchema~PageData</b>	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.
fillColor	String	A 6-character hexadecimal color string, including the # sign.
opacity	Number	The opacity of the mark, from 0 to 255.

Implements: [MarkSchema~Mark](#)

## FreehandAnnotation

The [FreehandAnnotation](#) object.

### Type:

- [MarkSchema~Mark](#)

### Properties:

Name	Type	Description
rectangle	<a href="#">MarkSchema~Rectangle</a>	The position of the mark on the page.
pageData	<a href="#">MarkSchema~PageData</a>	The size of the page that the mark is on.
color	String	A 6-character hexadecimal color string, including the # sign.
opacity	Number	The opacity of the mark, from 0 to 255.
path	String	An SVG-style path, using M, L, and C commands.
thickness	Number	The thickness of the border in pixels.

Implements: [MarkSchema~Mark](#)

## FreehandSignature

The [FreehandSignature](#) object.

- [MarkSchema~Mark](#)

**Properties:**

Name	Type	Description
rectangle	<a href="#">MarkSchema~Rectangle</a>	The position of the mark on the page.
pageData	<a href="#">MarkSchema~PageData</a>	The size of the page that the mark is on.
color	String	A 6-character hexadecimal color string, including the # sign.
horizontalAlignment	String	A value from <a href="#">PCCViewer.Mark.HorizontalAlignment</a> .
path	String	An SVG-style path, using M, L, and C commands.
thickness	Number	The thickness of the border in pixels.

Implements: [MarkSchema~Mark](#)

## HighlightAnnotation

The [HighlightAnnotation](#) object.

**Type:**

- [MarkSchema~Mark](#)

**Properties:**

Name	Type	Description
fillColor	String	A 6-character hexadecimal color string, including the # sign.
startIndex	Number	The character index of the start of the selection.
selectedText	String	The selected text.
textLength	Number	The length of the selected text.
lineGroups	Array. <a href="#">&lt;MarkSchema~LineGroup&gt;</a>	The individual line rectangles that make up

the selection.

Implements: [MarkSchema~Mark](#)

## ImageStampAnnotation

The [ImageStampAnnotation](#) object.

### Type:

- [MarkSchema~Mark](#)

### Properties:

Name	Type	Description
rectangle	<a href="#">MarkSchema~Rectangle</a>	The position of the mark on the page.
pageData	<a href="#">MarkSchema~PageData</a>	The size of the page that the mark is on.
imageDataUr1	String.<base64>	The base64 encoded image data.
imageId	String	The ID associated with the image.

Implements: [MarkSchema~Mark](#)

## ImageStampRedaction

The [ImageStampRedaction](#) object.

### Type:

- [MarkSchema~Mark](#)

### Properties:

Name	Type	Description
rectangle	<a href="#">MarkSchema~Rectangle</a>	The position of the mark on the page.
pageData	<a href="#">MarkSchema~PageData</a>	The size of the page that the mark is on.
imageDataUr1	String.<base64>	The base64

		encoded image data.
imageId	String	The ID associated with the image.

Implements: [MarkSchema~Mark](#)

## LineAnnotation

The [LineAnnotation](#) object.

### Type:

- [MarkSchema~Mark](#)

### Properties:

Name	Type	Description
pageData	<a href="#">MarkSchema~PageData</a>	The size of the page that the mark is on.
endPoint	<a href="#">MarkSchema~Point</a>	The end of the line.
startPoint	<a href="#">MarkSchema~Point</a>	The start of the line.
color	String	A 6-character hexadecimal color string, including the # sign.
endHeadType	String	A value from <a href="#">PCCViewer.Mark.LineHeadType</a> .
opacity	Number	The opacity of the mark, from 0 to 255.
thickness	Number	The thickness of the line in pixels.

Implements: [MarkSchema~Mark](#)

## PolylineAnnotation

The [PolylineAnnotation](#) object.

### Type:

- [MarkSchema~Mark](#)

### Properties:

pageData	<a href="#">MarkSchema~PageData</a>	The size of the page that the mark is on.
points	Array. < <a href="#">MarkSchema~Point</a> >	The array of points making up the line.
color	String	A 6-character hexadecimal color string, including the # sign.
opacity	Number	The opacity of the mark, from 0 to 255.
thickness	Number	The thickness of the line in pixels.

Implements: [MarkSchema~Mark](#)

## RectangleAnnotation

The [RectangleAnnotation](#) object.

### Type:

- [MarkSchema~Mark](#)

### Properties:

Name	Type	Description
rectangle	<a href="#">MarkSchema~Rectangle</a>	The position of the mark on the page.
pageData	<a href="#">MarkSchema~PageData</a>	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.
fillColor	String	A 6-character hexadecimal color string, including the # sign.

opacity	Number	The opacity of the mark, from 0 to 255.
---------	--------	---

Implements: [MarkSchema~Mark](#)

## RectangleRedaction

The [RectangleRedaction](#) object.

### Type:

- [MarkSchema~Mark](#)

### Properties:

Name	Type	Description
rectangle	<a href="#">MarkSchema~Rectangle</a>	The position of the mark on the page.
pageData	<a href="#">MarkSchema~PageData</a>	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.
fillColor	String	A 6-character hexadecimal color string, including the # sign.
fontColor	String	A 6-character hexadecimal color string, including the # sign.
reason	String	The redaction reason for this redaction mark.

## StampAnnotation

The [StampAnnotation](#) object.

**Type:**

- [MarkSchema~Mark](#)

**Properties:**

Name	Type	Description
rectangle	<a href="#">MarkSchema~Rectangle</a>	The position of the mark on the page.
pageData	<a href="#">MarkSchema~PageData</a>	The size of the page that the mark is on.
color	String	A 6-character hexadecimal color string, including the # sign.
label	String	The text to display inside the stamp.

Implements: [MarkSchema~Mark](#)

## StampRedaction

The [StampRedaction](#) object.

**Type:**

- [MarkSchema~Mark](#)

**Properties:**

Name	Type	Description
rectangle	<a href="#">MarkSchema~Rectangle</a>	The position of the mark on the page.
pageData	<a href="#">MarkSchema~PageData</a>	The size of the page that the mark is on.
label	String	The text to display inside the stamp.

Implements: [MarkSchema~Mark](#)

## StrikethroughAnnotation

**Type:**

- [MarkSchema~Mark](#)

**Properties:**

Name	Type	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. < <a href="#">MarkSchema~LineGroup</a> >	The individual line rectangles that make up the selection.

Implements: [MarkSchema~Mark](#)

## TextAnnotation

The [TextAnnotation](#) object.

**Type:**

- [MarkSchema~Mark](#)

**Properties:**

Name	Type	Description
rectangle	<a href="#">MarkSchema~Rectangle</a>	The position of the mark on the page.
pageData	<a href="#">MarkSchema~PageData</a>	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.
fillColor	String	A 6-character hexadecimal color string, including the # sign.
fontColor	String	A 6-character hexadecimal color string, including the # sign.
fontName	String	The name of the font to use for the mark.
fontSize	Number	The size of the font, in pixels.
fontStyle	Array.<String>	An array of values any from <a href="#">PCCViewer.Mark.FontStyles</a> .
maxLength	Number	The maximum number of characters allowed.
horizontalAlignment	String	A value from <a href="#">PCCViewer.Mark.HorizontalAlignment</a> .
text	String	The text of the mark.
opacity	Number	The opacity of the mark, from 0 to 255.

Implements: [MarkSchema~Mark](#)

## TextAreaSignature

The [TextAreaSignature](#) object.

### Type:

- [MarkSchema~Mark](#)

### Properties:

Name	Type	Description
rectangle	<a href="#">MarkSchema~Rectangle</a>	The position of the mark on the page.
pageData	<a href="#">MarkSchema~PageData</a>	The size of the page that the mark is on.
fontColor	String	A 6-character hexadecimal color string, including the # sign.
fontName	String	The name of the font to use for the mark.
fontStyle	Array.<String>	An array of values any from <a href="#">PCCViewer.Mark.FontStyles</a> .
horizontalAlignment	String	A value from <a href="#">PCCViewer.Mark.HorizontalAlignment</a> .

maxFontSize	Number	The maximum size of the font, in pixels.
maxLength	Number	The maximum number of characters allowed.
text	String	The text of the mark.

Implements: [MarkSchema~Mark](#)

## TextHyperlinkAnnotation

The [TextHyperlinkAnnotation](#) object.

### Type:

- [MarkSchema~Mark](#)

### Properties:

Name	Type	Description
fillColor	String	A 6-character hexadecimal color string, including the # sign.
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. <a href="#">&lt;MarkSchema~LineGroup&gt;</a>	The individual line rectangles that make up the selection.
href	String	The URL that the link points to.

Implements: [MarkSchema~Mark](#)

## TextInputSignature

**Type:**

- [MarkSchema~Mark](#)

**Properties:**

Name	Type	Description
rectangle	<a href="#">MarkSchema~Rectangle</a>	The position of the mark on the page.
pageData	<a href="#">MarkSchema~PageData</a>	The size of the page that the mark is on.
mask	<a href="#">MarkSchema~Mask</a>	An input mask that will be displayed in the mark to assist the user from inputting undesirable characters.
fontColor	String	A 6-character hexadecimal color string, including the # sign.
fontName	String	The name of the font to use for the mark.
maxLength	Number	The maximum number of characters allowed.
text	String	The text of the mark.
horizontalAlignment	String	A value from <a href="#">PCCViewer.Mark.HorizontalAlignment</a> .

Implements: [MarkSchema~Mark](#)

## TextRedaction

The [TextRedaction](#) object.

**Type:**

- [MarkSchema~Mark](#)

**Properties:**

Name	Type	Description
rectangle	<a href="#">MarkSchema~Rectangle</a>	The position of the mark on the page.
pageData	<a href="#">MarkSchema~PageData</a>	The size of the page that the mark is on.
fontColor	String	A 6-character hexadecimal color string, including the # sign.
fontName	String	The name of the font to use for the mark.
fontSize	Number	The size of the font, in pixels.
maxLength	Number	The maximum number of characters

		allowed.
horizontalAlignment	String	A value from <a href="#">PCCViewer.Mark.HorizontalAlignment</a> .
text	String	The text of the mark.

Implements: [MarkSchema~Mark](#)

## TextSelectionRedaction

The [TextSelectionRedaction](#) object.

### Type:

- [MarkSchema~Mark](#)

### Properties:

Name	Type	Description
lineGroups	Array. < <a href="#">MarkSchema~LineGroup</a> >	The individual line rectangles that make up the selection.
reason	String	The redaction reason for this redaction mark.
selectedText	String	The selected text.
startIndex	Number	The character index of the start of the selection.
textLength	Number	The length of the selected text.

Implements: [MarkSchema~Mark](#)

## TextSignature

The [TextSignature](#) object.

### Type:

- [MarkSchema~Mark](#)

### Properties:

rectangle	MarkSchema~Rectangle	The position of the mark on the page.
pageData	MarkSchema~PageData	The size of the page that the mark is on.
color	String	A 6-character hexadecimal color string, including the # sign.
fontName	String	The name of the font to use for the mark.
horizontalAlignment	String	A value from <a href="#">PCCViewer.Mark.HorizontalAlignment</a> .
text	String	The text of the mark.

Implements: [MarkSchema~Mark](#)

## TransparentRectangleRedaction

The [TransparentRectangleRedaction](#) object.

### Type:

- [MarkSchema~Mark](#)

### Properties:

Name	Type	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.

Implements: [MarkSchema~Mark](#)

## Comment

A mark comment.

### Type:

- Object

### Properties:

Name	Type	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	Type	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	Type	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	Type	Description
uid	String	A global unique ID for this mark, to identify it across the system.
type	String	A value from <a href="#">PCCViewer.Mark.Type</a> denoting the mark type.
pageNumber	Number	The page that the mark is located on.
creationDateTime	String	An ISO string of the created time.
modificationDateTime	String	An ISO string of the last modified time.
interactionMode	String	A value from <a href="#">PCCViewer.Mark.InteractionMode</a> .
data	Object.<string, string>	A property bag of user-defined values.
conversation	<a href="#">MarkSchema~Conversation</a>	The conversation object.

## Mask

An object defining the mask for this mark.

### Type:

- Object

### Properties:

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.

## PageData

An object providing metadata about the page at the time that the mark was saved.

### Type:

- Object

### Properties:

Name	Type	Description
width	Number	The width of the page at the time the mark was saved.
height	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	Type	Description
x	Number	The distance from the left edge of the page.
y	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

Name	Type	Description
x	Number	The left side of the rectangle, relative to the left page edge.
y	Number	The top of the rectangle, relative to the top page edge.
width	Number	The width of the bounding rectangle.
height	Number	The height of the bounding rectangle.

Documentation generated by *JSDoc 3.3.3* on Wed Mar 29 2017 10:18:08 GMT-0400 (Eastern Daylight Time)

## Namespace: MarkupLayerSchema

### 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	Type	Description
name	string	The name of the markup layer.
originalXmlName	string	The name of the web tier XML record from which the marks of this layer were originally stored.
data	string	A property bag of user-defined values.
marks	Array	An array of <a href="#">MarkSchema~Mark</a> objects. Note that comments on marks in this layer are stored in this comments array and not stored under the particular mark.
comments	Array	An array of <a href="#">MarkupLayerSchema~Comment</a> 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	Type	Description
markUid	string	The global unique ID for the mark the comment is under.
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.

---

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Namespace: MouseTools

### 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

[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.  <b>Note:</b> 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

The MouseTool object that was created.

Type

`PCCViewer.MouseTool`

### Example

```
// Create a new mouse tool with the
name "MyLineAnnotationMouseTool"
var myMouseTool =
PCCViewer.MouseTools.createMouseTool("
"LineAnnotation");

// Configure the mouse tool or the
template mark of the mouse tool
myMouseTool.getTemplateMark().setOpaci

// set the ViewerControl to use the
mouse tool
viewerControl.setCurrentMouseTool("MyL
```

(static) `getMouseTool(name)` →  
{`PCCViewer.MouseTool` | undefined}

Gets a named mouse tool.

### Parameters:

Name	Type	Description
name	string	The name of the MouseTool 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";
```

```
var mouseTool =
PCCViewer.MouseTools.getMouseTool(mous

// 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.LineAnnotatio

        ...
        break;
    default:
        ...
    }
}
```

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:45 GMT-0400 (Eastern Daylight Time)

## Namespace: PCCViewer

### Namespace: PCCViewer

#### PCCViewer

PCCViewer is the global namespace used for members of this API.

#### Classes

- [AjaxResponse](#)
- [BurnRequest](#)
- [Comment](#)
- [Conversation](#)
- [ConversionRequest](#)
- [DocumentHyperlink](#)
- [Error](#)
- [Event](#)

LoadMarkupLayersRequest

Mark

MarkupLayer

MarkupLayerCollection

MouseTool

ObservableCollection

PrintRequest

Promise

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:

ViewerReady	string	Triggered when the Viewer is ready. Augmented properties of the <b>PCCViewer.Event</b> object for this event: <ul style="list-style-type: none"><li>• None</li></ul>
PageCountReady	string	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. Augmented properties of the <b>PCCViewer.Event</b> object for this event: <ul style="list-style-type: none"><li>• <b>pageCount</b> {number} The actual page count of the document.</li></ul>
EstimatedPageCountReady	string	Event is triggered when the viewer has an estimated page count from the server. Augmented properties of the <b>PCCViewer.Event</b> object for this event: <ul style="list-style-type: none"><li>• <b>pageCount</b> {number} The estimated page count of the document.</li></ul>
PageChanged	string	Event is triggered when the viewer changed the current page. Augmented properties of the <b>PCCViewer.Event</b> object for this event: <ul style="list-style-type: none"><li>• none</li></ul>
PageLoadFailed	string	Event is triggered when the viewer changed the current page. Augmented properties of the <b>PCCViewer.Event</b> object for this event: <ul style="list-style-type: none"><li>• <b>pageNumber</b> {number} Indicates the page number of the page that failed to load.</li><li>• <b>statusCode</b> {number} Indicates the HTTP page load failure error code returned by the image service</li><li>• <b>accusoftErrorNumber</b> {number} The error codes in this category currently are:<ul style="list-style-type: none"><li>◦ 4001 Document requires a password (HTTP statusCode will be 480)</li><li>◦ 5001 Unable to generate Page (HTTP statusCode will be 580)</li><li>◦ 5002 Download of the file to the Image service failed (HTTP statusCode will be 580)</li></ul></li></ul>

		<ul style="list-style-type: none"> <li>• <code>accusoftErrorMessage</code> {string} Description of the error provided by the Image service.</li> </ul>
<code>PageDisplayed</code>	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 <code>PageDisplayed</code> event will fire for out-of-view pages.</p> <p>Augmented Properties of the <code>PCCViewer.Event</code> object for this event:</p> <ul style="list-style-type: none"> <li>• <code>pageNumber</code> {number} The page number of the displayed page.</li> </ul>
<code>PageRotated</code>	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 <code>PCCViewer.ViewerControl.Event</code> object for this event:</p> <ul style="list-style-type: none"> <li>• <code>pageNumber</code> {number} The page number of the page that was rotated.</li> </ul>
<code>DocumentRotated</code>	string	<p>Event is triggered when the rotation of all pages in the document changes.</p> <p>Augmented properties of the <code>PCCViewer.Event</code> object for this event:</p> <ul style="list-style-type: none"> <li>• <code>degreesClockwise</code> {number} The amount in degrees clockwise the pages were rotated.</li> </ul>
<code>ScaleChanged</code>	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 <code>PCCViewer.Event</code> object for this event:</p> <ul style="list-style-type: none"> <li>• <code>scaleType</code> {string} Gives an indication of whether the content was <code>scaledUp</code> (got bigger) or <code>scaledDown</code> (got smaller).</li> <li>• <code>scaleFactor</code> {number} Indicates the new scale factor of the viewer. A value of 1 indicates 100% zoom. See also</li> </ul>

		<p><a href="#">PCCViewer.ViewerControl#getScaleFactor</a>.</p> <ul style="list-style-type: none"> <li>• <code>trigger</code> {string} Indicates how the scale change was triggered. Possible values are: <ul style="list-style-type: none"> <li>◦ "Zoom" - Indicates a direct zoom, such as using the <a href="#">PCCViewer.ViewerControl#zoomIn</a>, <a href="#">PCCViewer.ViewerControl#zoomOut</a>, or <a href="#">PCCViewer.ViewerControl#setScaleFactor</a> methods.</li> <li>◦ "Fit" - Indicates a change due to a fit type being applied through the <a href="#">PCCViewer.ViewerControl#fitContent</a> method.</li> </ul> </li> <li>• <code>fitType</code> {string} Indicates the fit type that was applied, if applicable. This property will only be defined if the <code>trigger</code> was Fit, and will be undefined otherwise.</li> </ul>
DocumentPrinted	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 <a href="#">PCCViewer.Event</a> object for this event:</p> <ul style="list-style-type: none"> <li>• <code>pageNumbers</code> {Array.&lt;number&gt;} An array containing the page number of each page that was printed. <b>NOTE:</b> In the PageView viewer, the array contains the current page only.</li> <li>• <code>orientation</code> {string} "portrait" or "landscape"</li> <li>• <code>includeMarks</code> {boolean} Indicates whether the marks were included in the printed pages.</li> </ul>
TextSelected	string	<p>Event is triggered when text is selected.</p> <p>Augmented properties of the <a href="#">PCCViewer.Event</a> object for this event:</p> <ul style="list-style-type: none"> <li>• <code>selectedText</code> {string} Deprecated since v9.2 (use the <code>textSelection.text</code> argument instead).</li> <li>• <code>pageNumbers</code> {Array.&lt;number&gt;} Deprecated since v9.2 (use the <code>textSelection.pageNumber</code> argument instead).</li> <li>• <code>textSelection</code> {PCCViewer.ViewerControl.TextSelection} An object that provides information regarding the text selection.</li> </ul>

		<ul style="list-style-type: none"> <li>• <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.</li> <li>• <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.</li> <li>• <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 is stationary throughout the drag, and the other end is sliding.)</li> <li>• <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.)</li> </ul>
<code>MouseToolChanged</code>	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 <code>PCCViewer.Event</code> object for this event:</p> <ul style="list-style-type: none"> <li>• <code>mouseToolName</code> {string} Indicates the name of the new mouse tool.</li> </ul>
<code>SearchPerformed</code>	string	<p>Triggered when a search is performed with a call to <code>PCCViewer.ViewerControl#search</code>.</p> <p>Augmented properties of the <code>PCCViewer.Event</code> object for this event:</p> <ul style="list-style-type: none"> <li>• <code>searchRequest</code> {<code>PCCViewer.SearchRequest</code>} The search request returned from the call to <code>PCCViewer.ViewerControl#search</code>.</li> </ul>
<code>PartialSearchResultsAvailable</code>	string	<p>Event is triggered when partial search results are available</p> <p>Augmented properties of the <code>PCCViewer.Event</code> object for this event:</p>

		<ul style="list-style-type: none"> <li>• <code>partialSearchResults</code> {Array.&lt;<a href="#">PCCViewer.SearchResult</a>&gt;} The new search results found since the last "PartialSearchResultsAvailable" event.</li> <li>• <code>pagesWithoutText</code> {Array.&lt;number&gt;} 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.</li> </ul>
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 <a href="#">PCCViewer.Event</a> object for this event:</p> <ul style="list-style-type: none"> <li>• <code>completedSearchResults</code> {Array.&lt;<a href="#">PCCViewer.SearchResult</a>&gt;} The set of search results.</li> </ul>
SearchFailed	string	<p>Event is triggered when search failed to due to an exception.</p> <p>Augmented properties of the <a href="#">PCCViewer.Event</a> object for this event:</p> <ul style="list-style-type: none"> <li>• none</li> </ul>
SearchCancelled	string	<p>Event is triggered when search is cancelled by the user.</p> <p>Augmented properties of the <a href="#">PCCViewer.Event</a> object for this event:</p> <ul style="list-style-type: none"> <li>• <code>errorMessage</code> {string} A human readable error message indicating why the search failed.</li> </ul>
SearchResultsAvailable	string	<p>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.</p> <p>Augmented properties of the <a href="#">PCCViewer.Event</a> object for this event:</p> <ul style="list-style-type: none"> <li>• <code>completedSearchResults</code> {Array.&lt;<a href="#">PCCViewer.SearchResult</a>&gt;} The set of search results.</li> </ul>
SearchCleared	string	<p>Event is triggered when the current search is cleared. After this event, calls to <code>PCCViewer#setSelectedSearchResult</code>,</p>

		<p>PCCViewer#getSelectedSearchResult, and PCCViewer#getSearchRequest will no longer be valid.</p> <p>Augmented properties of the <b>PCCViewer.Event</b> object for this event:</p> <ul style="list-style-type: none"> <li>• none</li> </ul>
SearchResultSelectionChanged	string	<p>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.</p> <p>Augmented properties of the <b>PCCViewer.Event</b> object for this event:</p> <ul style="list-style-type: none"> <li>• none</li> </ul>
PrintRequested	string	<p>Event is triggered when a document print is requested through <b>PCCViewer.ViewerControl#print</b>.</p> <p>Augmented properties of the <b>PCCViewer.Event</b> object for this event:</p> <ul style="list-style-type: none"> <li>• <b>printRequest</b> {<b>PCCViewer.PrintRequest</b>}</li> </ul>
MarkupLoaded	string	<p>Event is triggered when annotations are loaded from a file. Triggered when annotations are loaded from a file.</p> <p>Augmented properties of the <b>PCCViewer.Event</b> object for this event:</p> <ul style="list-style-type: none"> <li>• <b>name</b> {string} The name of the markup data that was loaded.</li> <li>• <b>loadedMarks</b> {Array.&lt;<b>PCCViewer.Mark</b>&gt;} The marks that were loaded.</li> </ul>
MarkupSaved	string	<p>Event is triggered when annotations save to the server.</p> <p>Augmented properties of the <b>PCCViewer.Event</b> object for this event:</p> <ul style="list-style-type: none"> <li>• <b>name</b> {string} The name of the markup data that was saved.</li> </ul>
MarkChanged	string	<p>Event is triggered when one or more attributes changes on an annotation.</p> <p>Augmented properties of the <b>PCCViewer.Event</b> object for this event:</p> <ul style="list-style-type: none"> <li>• <b>mark</b> {<b>PCCViewer.Mark</b>} The changed annotation object</li> </ul>

	<ul style="list-style-type: none"><li>• <code>pageNumber</code> {number} The page number of the annotation.</li><li>• <code>propertyNames</code> {Array.&lt;string&gt;} The names of properties that have changed.</li></ul>
MarkCreated	<p>string Event is triggered when a new annotation is created.</p> <p>Augmented properties of the <code>PCCViewer.Event</code> object for this event:</p> <ul style="list-style-type: none"><li>• <code>mark</code> {<code>PCCViewer.Mark</code>} The annotation object.</li><li>• <code>pageNumber</code> {number} The page number of the annotation.</li><li>• <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 undefined for marks added using the API.</li><li>• <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 undefined for marks added using the API.</li></ul>
MarkRemoved	<p>string Event is triggered when a annotation is removed from a page.</p> <p>Augmented properties of the <code>PCCViewer.Event</code> object for this event:</p> <ul style="list-style-type: none"><li>• <code>mark</code> {<code>PCCViewer.Mark</code>} The annotation object.</li><li>• <code>pageNumber</code> {number} The page number of the annotation.</li></ul>
MarkReordered	<p>string Event is triggered when the annotation's stacking order has changed.</p> <p>Augmented properties of the <code>PCCViewer.Event</code> object for this event:</p> <ul style="list-style-type: none"><li>• <code>mark</code> {<code>PCCViewer.Mark</code>} The annotation object.</li><li>• <code>pageNumber</code> {number} The page number of the annotation.</li><li>• <code>index</code> {number} The new stacking order index of the annotation.</li><li>• <code>oldIndex</code> {number} The old stacking order index of the annotation.</li></ul>

MarkSelectionChanged	string	Triggered when the set of selected annotations has changed.  Augmented properties of the <b>PCCViewer.Event</b> object for this event: <ul style="list-style-type: none"><li>• <b>pageNumber</b> {number} The page number containing the mark that was selected or deselected.</li></ul>
MarkMouseEnter	string	Event is triggered when the mouse enters the annotation bounding box.  Augmented properties of the <b>PCCViewer.Event</b> object for this event: <ul style="list-style-type: none"><li>• <b>mark</b> {<b>PCCViewer.Mark</b>} The annotation object.</li><li>• <b>clientX</b> {number} A value indicating the absolute window position in the x-axis of the cursor.</li><li>• <b>clientY</b> {number} A value indicating the absolute window position in the y-axis of the cursor.</li></ul>
MarkMouseOver	string	Event is triggered when the mouse moves over the annotation bounding box.  Augmented properties of the <b>PCCViewer.Event</b> object for this event: <ul style="list-style-type: none"><li>• <b>mark</b> {<b>PCCViewer.Mark</b>} The annotation object.</li><li>• <b>clientX</b> {number} A value indicating the absolute window position in the x-axis of the cursor.</li><li>• <b>clientY</b> {number} A value indicating the absolute window position in the y-axis of the cursor.</li></ul>
MarkMouseLeave	string	Event is triggered when the mouse leaves the annotation bounding box.  Augmented properties of the <b>PCCViewer.Event</b> object for this event: <ul style="list-style-type: none"><li>• <b>mark</b> {<b>PCCViewer.Mark</b>} The annotation object.</li><li>• <b>clientX</b> {number} A value indicating the absolute window position in the x-axis of the cursor.</li><li>• <b>clientY</b> {number} A value indicating the absolute window position in the y-axis of the cursor.</li></ul>
CommentsPanelToggled	string	Triggered when the comments panel opens or

		<p>closes.</p> <p>Augmented properties of the <b>PCCViewer.Event</b> object for this event:</p> <ul style="list-style-type: none"><li>• <b>isOpen</b> {boolean} Whether the comments panel is open or closed.</li></ul>
CommentCreated	string	<p>Triggered when a comment is added to a Conversation in the viewer.</p> <p>Augmented properties of the <b>PCCViewer.Event</b> object for this event:</p> <ul style="list-style-type: none"><li>• <b>conversation</b> {PCCViewer.Conversation} The Conversation containing the changed comment.</li><li>• <b>comment</b> {PCCViewer.Comment} The comment that was added.</li></ul>
CommentRemoved	string	<p>Triggered when a comment is removed from a Conversation in the viewer.</p> <p>Augmented properties of the <b>PCCViewer.Event</b> object for this event:</p> <ul style="list-style-type: none"><li>• <b>conversation</b> {PCCViewer.Conversation} The Conversation containing the changed comment.</li><li>• <b>comment</b> {PCCViewer.Comment} The comment that was removed.</li></ul>
CommentChanged	string	<p>Triggered when the text of a comment in the viewer has changed.</p> <p>Augmented properties of the <b>PCCViewer.Event</b> object for this event:</p> <ul style="list-style-type: none"><li>• <b>conversation</b> {PCCViewer.Conversation} The Conversation containing the changed comment.</li><li>• <b>comment</b> {PCCViewer.Comment} The comment that was modified.</li></ul>
PageTextReady	string	<p>Triggered when the text of a page has been loaded in the viewer.</p> <p>Augmented properties of the <b>PCCViewer.Event</b> object for this event:</p> <ul style="list-style-type: none"><li>• <b>pageNumber</b> {number} The page number of the page that text is ready for.</li></ul>
Click	string	<p>Triggered when a user clicks a page or comment pane in the viewer.</p>

Augmented properties of the [PCCViewer.Event](#) object for this event:

- `pageNumber` {number} The page number of the page that was clicked, or null if none.
- `targetType` {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).
- `textSelection` {object} The text selection that was clicked, or null if none.
- `mark` {object} - The mark object that was clicked, or null if none.
- `searchResult` {object} - The search result that was clicked, or null if none.
- `documentHyperlink` {[PCCViewer.DocumentHyperlink](#)} - The document hyperlinks that was clicked, or null if none.
- `originalEvent` {object} - A copy of the browser event.
- `clientX` {number} - The x window coordinate of the position where the user clicked.
- `clientY` {number} - The y window coordinate of the position where the user clicked.

PageOpening

string 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.

Augmented properties of the [PCCViewer.Event](#) object for this event:

- `width` {number} The width in pixels of the page that is opening
- `height` {number} The height in pixels of the page that is opening
- `pageNumber` {number} The 1-based number of the page that is opening

See: [PCCViewer.ViewerControl#on](#)  
[PCCViewer.ViewerControl#off](#)

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:**

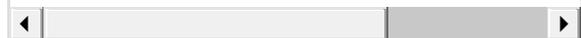
Name	Type	Description
FullWidth	string	The viewer scales the content to fill the width of the viewer.
ShrinkToWidth	string	The viewer will scale down the content until it fits fully width-wise into view. The page will not be scaled up if it already fits.
ActualSize	string	The viewer shows the content 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**

```
// us the enumeration
myViewerControl.fitContent(PCCViewer.F

// or just use the string value
myViewerControl.fitContent("FullWidth"
```



(static, readonly) `MarkHandleMode` :string

The `MarkHandleMode` enumeration defines mark handle

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	Type	Description
HideSideHandlesWhenClose	string	
HideCornerHandlesWhenClose	string	

See:

[PCCViewer.ViewerControl#getMarkHandleMode](#)

[PCCViewer.ViewerControl#setMarkHandleMode](#)

**Example**

```
// use the enumeration
myViewerControl.setMarkHandleMode(PCCV

// or just use the string value
myViewerControl.setMarkHandleMode("Hid
```

(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:**

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(PCCViewe

// or just use the string value
myViewerControl.setPageLayout("Horizon
```

(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	Type	Description
Draft	string	The viewer displays the document content underneath the redaction rectangles in the document.
Normal	string	The viewer hides the document content underneath the redaction rectangles.

See: [PCCViewer.ViewerControl#getRedactionViewMode](#)

### Example

```
// use the enumeration
myViewerControl.setRedactionViewMode(P

// or just use the string value
myViewerControl.setRedactionViewMode("


```

(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	Type	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.

view mode (enumeration values) directly to the API.

**Type:**

- string

**Properties:**

Name	Type	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.
```



Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:44 GMT-0400 (Eastern Daylight Time)

## Namespace: Signatures

### 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	Type	Attributes	Description
type	string		Describes the type of data in this signature. For FreehandSignature, this value will always be path.
path	string		The path data of the signature.
width	number		The absolute width of the path data.
height	number		The absolute height of the path data.
category	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	Type	Attributes	Description
type	string		Describes the type of data in this signature. For TextSignature, this value will always be text.
text	string		The text contents of the signature.
category	string   undefined	<optional>	The category of this signature. See signatureCategories in <a href="#">external:jQuery.fn~Options</a> for more information.

---

Documentation generated by [JSDoc 3.3.3](#) on Mon Mar 13 2017 16:48:45 GMT-0400 (Eastern Daylight Time)

## Namespace: Util

### 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

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:***

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"><li>• <b>startIndex {number}</b> - required<ul style="list-style-type: none"><li>◦ The start index of the selection, in a 0-based index of string characters.</li><li>◦ The valid range is <code>startIndex &gt;= 0</code>.</li></ul></li><li>• <b>length {number}</b> - required<ul style="list-style-type: none"><li>◦ The length of the selection, in characters.</li><li>◦ The valid range is <code>length &gt; 0</code>.</li></ul></li><li>• <b>color {string}</b> - required<ul style="list-style-type: none"><li>◦ Specifies the Hexadecimal color for the selection.</li><li>◦ 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.</li></ul></li><li>• <b>opacity {number}</b> - required<ul style="list-style-type: none"><li>◦ Specifies the opacity of the</li></ul></li></ul>
------------	--------------------------------	---

	selection. <ul style="list-style-type: none"><li>Valid values are from 0 to 255 (inclusive).</li></ul>
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

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, options<sub>opt</sub>)** → {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		A string specifying page numbers or ranges. Valid values are any string using this format (specified using EBNF): <pre>range      = "all"   pageRange; pageRange = pageNumber, [",", range]              subRange, [",", range]; pageNumber = naturalNumber; subRange   = pageNumber, "-", pageNumber;</pre> For example: "all", "1,2,3", and "1, 5-10" are all valid ranges.		
options	object	<optional>	An object specifying validation options.		
<b>Properties</b>					
	<b>Name</b>	<b>Type</b>	<b>Attributes</b>	<b>Default</b>	<b>Description</b>
	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.
<code>allowEmpty</code> 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  
3-5"); // returns [1, 3, 4, 5]  
  
PCCViewer.Util.convertPageRangeToArray  
3-5", {  
    lowerLimit: 1,  
    upperLimit: 6  
}); // returns [1, 3, 4, 5]
```

```
PCCviewer.UC11.ConvertPageRangeToArray  
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:**

colors	<p>Array. &lt;Object&gt;   Object</p> <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"><li>• color {string} - required<ul style="list-style-type: none"><li>◦ Specifies the Hexadecimal color for the highlight.</li><li>◦ 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.</li></ul></li><li>• opacity {number} - required<ul style="list-style-type: none"><li>◦ Specifies the opacity of the highlight.</li><li>◦ Valid values are from 0 to 255 (inclusive).</li></ul></li></ul>
backgroundColor	<p>string</p> <p>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.</p>

### 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<sub>opt</sub>)** →  
{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		A string specifying page numbers or ranges. Valid values are any string using this format (specified using EBNF): <pre>range      = "all"   pageRange; pageRange = pageNumber, [",", range]              subRange, [",", range]; pageNumber = naturalNumber; subRange   = pageNumber, "-", pageNumber;</pre> For example: "all", "1,2,3", and "1, 5-10" are all valid ranges.
options	object	<optional>	An object specifying validation options.

Properties				
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
```

## PrizmDoc Application Services (PAS) RESTful API

### Developer Reference

This section contains the following information:

- PrizmDoc Application Services (PAS) RESTful API
  - [Developer Reference](#)
  - [API Data Types](#)
  - [Back End Proxy](#)
  - [Content Converters](#)
  - [Content Converters Deprecated](#)
  - [Form Definitions](#)
  - [Form Extractors](#)
  - [Health](#)
  - [Image Stamps](#)
  - [Legacy Create Session](#)
  - [Markup Burner](#)
  - [Markup Layers](#)
  - [Markup XML](#)
  - [Search Tasks](#)
  - [Viewing Package Creators](#)
  - [Viewing Packages](#)
  - [Viewing Session](#)

### API Data Types

#### API Data Types

PrizmDoc 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:

<b>number</b>	Any number. This includes numbers with or without decimals.	1000.15 or 1500
<b>integer</b>	whole numbers only	120
<b>boolean</b>	true or false (without quotes)	true or false
<b>date</b>	This is the <a href="#">ISO 8601</a> profile for the <b>full-date</b> as described in the <a href="#">RFC 3339 section 5.6, Internet Date/Time Format</a> . The syntax for <b>full-date</b> as described in this document is as full-date = YYYY(4 digits) "-" MM(01 through 12) "-" DD(01 through 31)	2015-05-12
<b>dateTime</b>	This is the <a href="#">ISO 8601</a> profile for the <b>date-time</b> as described in the <a href="#">RFC 3339 section 5.6, Internet Date/Time Format</a> . The <b>date-time</b> syntax described in this document is 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	November 5, 2015, 8:15:30 am, US Eastern Standard Time : 2015-11-05T08:15:30-05:00 Same instant in UTC : 2015-11-05T13:15:30Z

	<p>time zone offsets:</p> <ol style="list-style-type: none"><li>1. Times are expressed in UTC (Coordinated Universal Time), with a special UTC designator ("Z").</li><li>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 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.</li></ol>	
<b>object</b>	A JSON object	<code>{"fileName": "sample.doc"}</code>
<b>array</b>	An array object	<code>["one", "two", "three"]</code>
<b>string</b>	A sequence of zero	<code>"abcdefhh"</code>

	or more characters	
<b>url</b>	A string which is a URL	"http://example.com"
<b>urlSafeBase64</b>	A URL-safe base64 encoded string, according to <a href="#">RFC 4648 Section 5</a>	"Pqu_fKOCYd1QM5oJW6pz-suKQ-2fuxbdZtCKcApvMFVP9GGKv99crwyXTr6AZjrC5vvi3acnZVLgyEXzA"

## Back End Proxy

### Back-end Proxy

The following routes are proxied to the configured PrizmDoc Server through PrizmDoc Application Services. More information about these routes, their work, and their responses can be found in the PrizmDoc Server REST API documentation.

#### Routes for document viewing

GET /License/ClientViewer

**Routes key:** `GetClientViewerLicense`

```
GET http://localhost:3000/License/ClientViewer
```

GET /Document/q/Attributes?DocumentID=u{viewingSessionId}

**Routes key:** `GetDocumentAttributes`

```
GET http://localhost:3000/Document/q/Attributes?DocumentID=u1234
```

GET /Document/q/{pageNumber}/Text?DocumentID=u{viewingSessionId}

**Routes key:** `GetPageText`

```
GET http://localhost:3000/Page/q/0/Text?DocumentID=u1234
```

GET /Page/q/{pageNumber}?DocumentID=u{viewingSessionId}

**Routes key:** `GetPage`

```
GET http://localhost:3000/Page/q/0/?DocumentID=u1234&ContentType=svg
```

GET /Page/q/{pageNumber}/Tile/{x}/{y}/{width}/{height}?DocumentID=u{viewingSessionId}

**Routes key: `GetPageTile`**

```
GET http://localhost:3000/Page/q/0/Tile/0/0/256/256?DocumentID=u1234
```

GET /Page/q/{pageNumber}/Attributes?DocumentID=u{viewingSessionId}

**Routes key: `GetPageAttributes`**

```
GET http://localhost:3000/Page/q/0/Attributes?DocumentID=u1234
```

GET /Page/q/{pageNumber}/{width}x{height}?DocumentID=u{viewingSessionId}

**Routes key: `GetThumbnail`**

```
GET http://localhost:3000/Page/q/0/200x200?DocumentID=u1234
```

## 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.

```
POST http://localhost:3000/ViewingSession/u1234/Replacement
Content-Type: application/json
{ "password": "pdfPassword" }
```

GET /ViewingSession/u{viewingSessionId}/SourceFile

**Routes key: `GetSourceFile`**

Downloads the original document that is being viewed.

```
GET http://localhost:3000/ViewingSession/u1234/SourceFile
```

For legacy reasons, a second route is available for downloading the original document, as such:

**GET /SaveDocument/q?DocumentID=u{viewingSessionId}**

**Routes key: `SaveDocument`**

```
GET http://localhost:3000/SaveDocument/q?DocumentID=u1234
```

*Note: when downloading the document programmatically, it is suggested to use the `ViewingSession` based API instead of the `SaveDocument` legacy API.*

```
GET /ViewingSession/u{viewingSessionId}/Attachments
```

**Routes key:** `GetAttachments`

Gets information about the document attachments (such as ones available on an `eml` or `msg` file).

```
GET http://localhost:3000/ViewingSession/u1234/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.

```
POST http://localhost:3000/ViewingSession/u1234/MarkupBurner
Content-Type: application/xml
<?xml version="1.0">...
```

```
GET /ViewingSession/u{viewingSessionId}/MarkupBurner/{markupBurnerId}
```

**Routes key:** `Pol1MarkupBurner`

Checks the status of the markup burning task.

```
GET http://localhost:3000/ViewingSession/u1234/MarkupBurner/abcd
```

```
GET /ViewingSession/u{viewingSessionId}/MarkupBurner/{markupBurnerId}/Document
```

**Routes key:** `GetBurnedDocument`

Downloads the resulting burned-in document.

```
GET http://localhost:3000/ViewingSession/u1234/MarkupBurner/abcd/Document
```

## Content Converters

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.
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. For the initial POST, this value will almost always be "processing". Results are typically only available with a subsequent GET.
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.

## Examples

POST http://localhost:3000/v2/viewingSessions/ZGZhc2RmYXNkZmFzZGZkcw/contentConverters

## Successful Response:

```
200 OK
Content-Type: application/json
{
  "input": {
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    },
    "sources": [
      {
        "pages": ""
      }
    ]
  },
  "expirationDateTime": "2015-11-04T19:20:09.280Z",
  "processId": "mxivIVSw7UhtL1yWwt3QEA",
  "state": "processing",
  "percentComplete": 0
}
```

## Errored Responses:

When a response with a status code of 580 is received check the response body for details:

```
580 InternalError
Content-Type: application/json
{
  "errorCode": "InternalError"
}
```

When a viewing session does not exist for the given `viewingSessionId`:

```
404 Not Found
```

When any data is present in the body of the request:

```
480 ReservedInput
{
  "errorCode": "ReservedInput",
  "errorDetails": {
    "in": "body"
  }
}
```

## GET /v2/viewingSessions/{viewingSessionId}/contentConverters/{processId}

### Routes key: Po11ContentConverterV2

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 GET /v2/viewingSessions/{viewingSessionId}/contentConverters/{processId}/results/{index}/file 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.

## Examples

```
GET http://localhost:3000/v2/viewingSessions/ZGZhc2RmYXNkZmFzZGZkcw/contentConverters/mxivIVSw7UhtL1yWwt3QEA
```

## Successful Response:

When the converter process completes with no conversion failures:

```
200 OK
Content-Type: application/json
{
  "input": {
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    },
    "sources": [
      {
        "pages": ""
      }
    ]
  },
  "processId": "mxivIVSw7UhtL1yWwt3QEA",
  "state": "complete",
  "percentComplete": 100,
  "output": {
    "results": [
      {
        "fileId": "e0sJIqI8aHkxVV0yJug",
        "sources": [
          {
            "pages": "1-4"
          }
        ],
        "pageCount": 4
      }
    ]
  }
}
```

When the converter process completes with a failure:

```
200 OK
Content-Type: application/json
{
  "input": {
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    },
    "sources": [
      {
        "pages": ""
      }
    ]
  },
  "processId": "mxivIVSw7UhtL1yWwt3QEA",
```

```
"errorCode": "CouldNotConvert",
"output": {
  "results": [
    {
      "errorCode": "CouldNotConvertPage",
      "sources": [
        {
          "pages": "3"
        }
      ]
    }
  ]
}
```

### Errored Responses:

When a response with a status code of 580 is received check the response body for details:

```
580 InternalError
Content-Type: application/json
{
  "errorCode": "InternalError"
}
```

When a converter process does not exist for the given `processId`:

```
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 POST <code>/ViewingSession</code> .
<code>processId</code>	The ID provided in the response from POST <code>/v2/viewingSessions/{viewingSessionId}/contentConverters</code> .
<code>index</code>	The 0-based index of the result file, originating from its position in the response.output.results array in the response from GET <code>/v2/viewingSessions/{viewingSessionId}/contentConverters/{processId}</code> .
<code>fileName</code>	The value that will be set for the Content-Disposition: filename header property in the response.

### Examples

```
GET http://localhost:3000/v2/viewingSessions/ZGZhc2RmYXNkZmFzZGZkcw/contentConverters/mxivIVSw7UhtL1yWwt3QEa/results/0/file?ContentDispositionFilename=MyFile
```

### Successful Response:

```
200 OK
Content-Disposition: attachment; filename=MyFile.pdf
Content-Type: {content type of the specific document}
```

### Errored Responses:

When an output file does not exist for the given `index`:

```
404 Not Found
```

## Content Converters Deprecated

### Content Converters (Deprecated)

**Note:** The following URLs have been deprecated and will be removed from the public API in a future release. Please use the new **Content Converters API** instead.

#### POST /contentConverters

##### 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:

```
200 OK
Content-Type: application/json
{
  "input": {
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    },
    "sources": [
      {
        "fileId": "9BgnvnYFK96E0Y0sK-A9xA",
        "pages": ""
      }
    ]
  },
  "expirationDateTime": "2015-11-04T19:20:09.280Z",
  "processId": "mxivIVSw7UhtL1yWwt3QEA",
  "state": "processing",
  "percentComplete": 0,
  "affinityToken": "wxyz"
}
```

##### Errored Responses:

```
580 Server Error
Content-Type: application/json
{ "errorCode": "InternalError" }
```

GET /contentConverters/{processId}

**Routes key:** `PollContentConverter`

Gets the status of the specific conversion task.

```
GET http://localhost:3000/contentConverters/9BgnvnYFK96E0Y0sK-A9xA
Accusoft-Affinity-Token: wxyz
```

**Successful Response:**

```
200 OK
Content-Type: application/json
{
  "input": {
    "dest": {
      "format": "pdf",
      "pdfOptions": {
        "forceOneFilePerPage": false
      }
    },
    "sources": [
      {
        "fileId": "9BgnvnYFK96E0Y0sK-A9xA",
        "pages": ""
      }
    ]
  },
  "expirationDateTime": "2015-11-04T19:20:09.280Z",
  "processId": "mxivIVSw7UhtL1yWwt3QEA",
  "state": "complete",
  "percentComplete": 100,
  "output": {
    "results": [
      {
        "fileId": "e0sJIqI8aHkxVW0yJug",
        "sources": [
          {
            "fileId": "9BgnvnYFK96E0Y0sK-A9xA",
            "pages": "1-4"
          }
        ]
      }
    ]
  }
}
```

```
    ],  
    "pageCount": 4  
  }  
]  
}  
}
```

## Errored Responses:

When an unknown error occurs while gathering data:

```
580 Server Error  
Content-Type: application/json  
{ "errorCode": "InternalError" }
```

GET /WorkFile/{fileId}?ContentDispositionFilename={file name}&affinityToken={affinityToken}

## Routes key: `GetWorkFile`

```
GET http://localhost:3000/WorkFile/e0sJIqI8aHkxVV0yJug?  
ContentDispositionFilename=MyFile&affinityToken=wxyz
```

## Successful Response:

```
200 OK  
Content-Disposition: attachment; filename={documentDisplayName}.{ext}  
Content-Type: {content type of the specific document}
```

## Form Definitions

### Form Definitions

GET /FormDefinitions

## Routes key: `GetFormDefinitions`

Gets the list of forms available on the server.

```
GET http://localhost:3000/FormDefinitions
```

### Successful Response:

```
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": "04a6032f3eaa4a8a9eb1b5fce1cb99e9"
  }
]
```

### Errored Response:

When an unknown error occurs while gathering data:

```
580 Server Error
Content-Type: application/json
{ "errorCode": "InternalError" }
```

GET /FormDefinitions/{formDefinitionId}

### Routes key: GetFormDefinition

Gets a specific form definition from the server.

```
http://localhost:3000/FormDefinitions/03f3e9c4a976419da576276acc427700
```

```
200 OK
Content-Type: application/json
{
  "templateDocumentId": "Form 1.pdf",
  "globalSettings": { ... global settings ... },
  "formRoles": { ... form roles ... },
  "groups": {},
  "formName": "Form 1",
  "formData": [ ... form data ... ]
}
```

## Errored Responses:

When the form definition does not exist:

```
404 Not Found
Content-Type: application/json
{ "errorCode": "NotFound" }
```

When an unknown error occurs while gathering data:

```
500 Server Error
Content-Type: application/json
{ "errorCode": "InternalError" }
```

## POST /FormDefinitions

### Routes key: **CreateFormDefinition**

Creates a new form definition using the uploaded data.

```
POST http://localhost:3000/FormDefinitions
Content-Type: application/json
{
  "templateDocumentId": "Form 3.pdf",
  "globalSettings": { ... global settings ... },
  "formRoles": { ... form roles ... },
  "groups": {},
  "formName": "Form 3",
  "formData": [ ... form data ... ]
}
```

```
201 Created
Content-Type: application/json
{ "formDefinitionId": "5418c96283bc469783bd30e7c8fdc059" }
```

## Errored Responses:

When an unknown error occurs while gathering data:

```
580 Server Error
Content-Type: application/json
{ "errorCode": "InternalError" }
```

## 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 http://localhost:3000/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:

```
200 OK
```

## Errored Responses:

When the form definition does not exist:

```
404 Not Found
Content-Type: application/json
{ "errorCode": "NotFound" }
```

When an unknown error occurs while gathering data:

```
580 Server Error
Content-Type: application/json
{ "errorCode": "InternalServerError" }
```

### DELETE /FormDefinitions/{formDefinitionId}

**Routes key:** DeleteFormDefinition

Deletes a form definition from the server.

```
DELETE http://localhost:3000/FormDefinitions/03f3e9c4a976419da576276acc427700
```

Alternatively, the POST method is supported for this request in combination with an X-HTTP-Method-Override header, as such:

```
POST http://localhost:3000/FormDefinitions/03f3e9c4a976419da576276acc427700
X-HTTP-Method-Override: DELETE
```

**Successful Response:**

```
204 No Content
```

**Errored Responses:**

When the form definition does not exist:

```
404 Not Found
Content-Type: application/json
{ "errorCode": "NotFound" }
```

When an unknown error occurs while gathering data:

```
580 Server Error
Content-Type: application/json
{ "errorCode": "InternalServerError" }
```

## Form Extractors

The *form extractors* API allows you to detect form field elements in viewing session source documents.

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
{viewingSessionId}	The <code>viewingSessionId</code> which identifies the viewing session. <b>Note this particular URL requires a letter 'u' to be provided before the viewingSessionId.</b>

#### 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 errorCode	Description
404		No viewing session with the provided {viewingSessionId} could be found.
480	"DocumentNotProvidedYet"	A source document has not been provided to the viewing session.
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

```
GET /ViewingSession/uDLbVh9sTmXJAm1GeXbS9Gn3WHxs8oib2xPsw2xEFjnIDdoJcudPtxciodSYFQq6zYGabQ_rJIecdbkImTTkSA/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.

Name	Description
Content-Type	Must be application/json

## 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 errorCode	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	"DocumentNotProvidedYet"	A source document has not been provided to the viewing session.
480	"FeatureNotLicensed"	You are not licensed to use the form extraction feature.
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 /v2/viewingSessions/uDLbVh9sTmXJAmD1GeXbS9Gn3WHxs8oib2xPsw2xEfjnIDdoJcudPtxciodSYFQq6zYGabQ_rJIecdbkImTTkSA/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.

Parameter	Description
{viewingSessionId}	The viewingSessionId which identifies the viewing session.
{processId}	The processId 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 errorCode	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.
500	"InternalError"	The server encountered an internal error when handling the request.

## Examples

### Request

```
GET
/v2/viewingSessions/uDLbVh9sTmXJAmD1GeXbS9Gn3WHxs8oib2xPsw2xEFjnIDdoJcudPtxciodSYFQq6zYGabQ_rJIecdbkImTTkSA/formExtractors/x62gH3TYdq1Kj94pLqzmts
```

### 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
              }
            }
          ]
        }
      ]
    }
  }
}
```

```
      "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": "x62gH3TYdqLk94pLqzmtS",
"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:

- "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 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.

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.

## Health

### Health and Information

The following are APIs for checking the health status of the PrizmDoc Application Services, as well as checking whether it is properly configured to connect with a PrizmDoc Server. Additionally an API is provided to return information about the PrizmDoc Application Services.

*Note that these routes do not have route keys, as they cannot be re-routed.*

### GET /health

A check to determine if the service is up and running.

```
GET http://localhost:3000/health
```

### Successful Response:

```
200 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 the PrizmDoc Application Services connectivity to PrizmDoc Server, whether self-hosted or configured through an Accusoft Cloud-Hosted PrizmDoc Server.

#### Successful Response:

```
200 OK
OK
```

*Note that this response represents that the connection to the back-end service is successful, and does not take into account whether those services are healthy. If you need to check the health of those services, please make a call to them directly.*

#### Errored Responses:

```
580
```

*This response represents that PAS is not properly configured to communicate with a Prizm back-end service.*

### GET /info

A request to get information about the service.

```
GET http://localhost:3000/info
```

#### Successful Response:

```
200 OK
Content-Type: application/json
{
  "version": "X.X.XXXX.XXXX"
}
```

Where the version is the PrizmDoc Application Service 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.

## Image Stamps

### Image Stamps

#### GET /ImageStampList

Routes key: `GetImageStamps`

```
GET http://localhost:3000/ImageStampList
```

#### Successful Response:

```
200 OK
Content-Type: application/json
{
  imageStamps : [
    { id: "Zm1sZTEuZ21m", displayName: "file1.gif" },
    { id: "Zm1sZTIucG5n", displayName: "file2.png" }
  ]
}
```

#### GET /ImageStamp/{imageStampId}/q?format={formatStr}

Routes key: `GetImageStamp`

Gets the image data from the server. 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.

Getting Base64 data:

```
GET http://localhost:3000/ImageStamp/Zm1sZTIucG5n/q?format=Base64
```

#### Successful Response:

```
200 OK
Content-Type: application/json
{
  "dataHash": "1ca8d2e80b6f8f2774f3bc0e6422bc653b0e4d80",
  "dataUrl": "data:image/png;base64,..."
}
```

Getting the image data:

```
GET http://localhost:3000/ImageStamp/Zm1sZTIucG5n/q?format=Image
```

## Successful Response:

```
200 OK
Content-Type: {the correct content type for the image}
{binary image data}
```

## Errored Responses:

When an invalid `imageStampId` is requested:

```
400 Bad Request
Content-Type: application/json
{
  "errorCode": "BadRequest"
}
```

When a valid `imageStampId` is requested but it does not exist:

```
404 Not Found
Content-Type: application/json
{
  "errorCode": "NotFound"
}
```

## Legacy Create Session

### Create Session (Legacy API)

The Legacy `Create Session` API is an API that matches the implementation of the `/CreateSession` endpoint of the previous Web Tier samples. This API is deprecated in PrizmDoc Application Services, and is available for backwards compatibility only. Please use the new [ViewingSession](#) API instead.

```
GET /CreateSession?document={document name}
```

Creates a viewing session based on a specific document in storage.

```
GET http://localhost:3000/CreateSession?document=Sample.doc
```

To make the equivalent call using the new `ViewingSession` API:

```
POST http://localhost:3000/ViewingSession
Content-Type: application/json
{
  "source": {
    "type": "document",
    "fileName": "Sample.doc"
  }
}
```

### Successful Response:

```
200 OK
Content-Type: application/json
{ "viewingSessionId": "1234..." }
```

### Errored Responses:

When the document does not exist:

```
404 Not Found
Content-Type: application/json
{ "errorCode": "DocumentNotFound" }
```

When an unknown error occurs while gathering data:

```
500 Server Error
Content-Type: application/json
{ "errorCode": "InternalError" }
```

GET /CreateSession?form={formDefinitionId}

### Routes key: `LegacyCreateSession`

Creates a viewing session for the document referenced by a form definition.

```
GET http://localhost:3000/Createsession?form=03t3e9c4a9/6419da5/62/6acc42/00
```

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.

### Errored Responses:

When the form definition does not exist:

```
404 Not Found
Content-Type: application/json
{ "errorCode": "FormDefinitionNotFound" }
```

When the document referenced by the form definition does not exist:

```
404 Not Found
Content-Type: application/json
{ "errorCode": "DocumentNotFound" }
```

When an unknown error occurs while gathering data:

```
500 Server Error
Content-Type: application/json
{ "errorCode": "InternalError" }
```

## Markup Burner

### Markup Burner

POST /ViewingSession/u{viewingSessionId}/MarkupBurner

#### Routes key: `CreateMarkupBurner`

Burn annotations and redactions into the document that is currently being viewed. This endpoint creates a burn task and returns immediately, often before the task is completed. Check the `state` value of the response to see the current state.

```
POST http://localhost:3000/ViewingSession/u1234
Content-Type application/xml
```

<HTML VERSION= 1.0 /...>

## Successful Response:

```
200 OK
Content-Type: application/json
{
  "input": null,
  "processId": "Be3enRqsA0ttkYwK1w5XOA",
  "state": "processing",
  "percentComplete": 0,
  "errorCode": null,
  "output": null
}
```

GET /ViewingSession/u{viewingSessionId}/MarkupBurner/{markupBurnerId}

## Routes key: `Pol1MarkupBurner`

Checks the status of the burn task. Check the `state` value of the response to see if it has completed or not. A completed task will have a `state` of `complete`.

```
GET http://localhost:3000/ViewingSession/u1234/MarkupBurner/Be3enRqsA0ttkYwK1w5XOA
```

## Successful Response:

```
200 OK
Content-Type: application/json
{
  "input": null,
  "processId": "Be3enRqsA0ttkYwK1w5XOA",
  "state": "complete",
  "percentComplete": 100,
  "errorCode": null,
  "output": null
}
```

GET

/ViewingSession/u{viewingSessionId}/MarkupBurner/{markupBurnerId}/Document

## Routes key: `GetBurnedDocument`

Downloads the resulting burned-in document. It will be served with the correct `Content-Type` of the

extension.

```
GET
http://localhost:3000/ViewingSession/u1234/MarkupBurner/Be3enRqsA0ttkYwK1w5X0A/Document?
ContentDispositionFilename={documentDisplayName}
```

## Successful Response:

```
200 OK
Content-Disposition: attachment; filename={documentDisplayName}.{ext}
Content-Type: {content type of the specific document}
```

## Markup Layers

### Markup Layers

GET /MarkupLayers/u{viewingSessionId}

#### Routes key: GetMarkupLayers

Gets the list of all available markup layers for the particular document.

```
GET http://localhost:3000/MarkupLayers/u1234
```

## Successful Response:

```
200 OK
Content-Type: application/json
[
  {
    "name": "layer name 1",
    "layerRecordId": "2fee93fadf2ed11df",
    "originalXmlName": ""
  },
  {
    "name": "layer name 2",
    "layerRecordId": "32f993b09fb0f2236",
    "originalXmlName": ""
  }
]
```

## Errored Responses:

When the document cannot be identified based on the viewing session:

```
502 Bad Gateway
Content-Type: application/json
{ "errorCode": "BadGateway" }
```

When an unknown error occurs while gathering data:

```
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 http://localhost:3000/MarkupLayers/u1234
Content-Type: application/json
{
  "name": "layer name 1",
  "comments": [],
  "data": {},
  "marks": [{ array of mark objects }]
}
```

## Successful Response:

```
201 Created
Content-Type: application/json
{ "layerRecordId": "2fee93fadf2ed11df" }
```

## Errored Responses:

When the document cannot be identified based on the viewing session:

```
502 Bad Gateway
Content-Type: application/json
{ "errorCode": "BadGateway" }
```

When an unknown error occurs while gathering data:

```
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 http://localhost:3000/MarkupLayers/u1234/2fee93fadf2ed11df
```

**Successful Response:**

```
200 OK
Content-Type: application/json
{ the markup layer data }
```

**Errored Responses:**

When the markup layer record does not exist:

```
404 Not Found
Content-Type: application/json
{ "errorCode": "NotFound" }
```

When the document cannot be identified based on the viewing session:

```
502 Bad Gateway
Content-Type: application/json
{ "errorCode": "BadGateway" }
```

When an unknown error occurs while gathering data:

```
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 http://localhost:3000/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 http://localhost:3000/MarkupLayers/u1234/2fee93fadf2ed11df
Content-Type: application/json
X-HTTP-Method-Override: PUT
```

## Successful Response:

```
200 OK
```

## Errored Responses:

When the markup layer record does not exist:

```
404 Not Found
Content-Type: application/json
{ "errorCode": "NotFound" }
```

When the document cannot be identified based on the viewing session:

```
502 Bad Gateway
Content-Type: application/json
{ "errorCode": "BadGateway" }
```

When an unknown error occurs while gathering data:

```
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 http://localhost:3000/MarkupLayers/u1234/2fee93fadf2ed11df
```

Alternatively, the POST method is supported for this request in combination with an X-HTTP-Method-Override header, as such:

```
POST http://localhost:3000/MarkupLayers/u1234/2fee93fadf2ed11df
X-HTTP-Method-Override: DELETE
```

### Successful Response:

```
204 No Content
```

### Errored Responses:

When the markup layer record does not exist:

```
404 Not Found
Content-Type: application/json
{ "errorCode": "NotFound" }
```

When the document cannot be identified based on the viewing session:

```
502 Bad Gateway
Content-Type: application/json
{ "errorCode": "BadGateway" }
```

When an unknown error occurs while gathering data:

```
580 Server Error
Content-Type: application/json
{ "errorCode": "InternalError" }
```

## Markup XML

### Markup XML

GET /AnnotationList/q/Art/q?DocumentID=u{viewingSessionId}

**Routes key: `GetAnnotations`**

Gets the list of available annotation XML files from the server.

```
GET http://localhost:3000/AnnotationList/q/Art/q?DocumentID=u1234
```

**Successful Response:**

```
200 OK
Content-Type: application/json
{
  "annotationFiles": [
    {
      "annotationLabel": "anId",
      "annotationName": "abcd_0_anId.xml",
      "ID": "1"
    }
  ]
}
```

GET /Document/q/Art/q?

DocumentID=u{viewingSessionId}&AnnotationID=u{annotationId}

**Routes key: `GetAnnotation`**

Gets a specific annotations XML file from the server.

```
GET http://localhost:3000/Document/q/Art/q?DocumentID=u1234&AnnotationID=uanId
```

**Successful Response:**

```
200 OK
Content-Type: application/xml
<?xml version="1.0"?>...
```

DocumentID=u{viewingSessionId}&AnnotationID=u{annotationId}

## Routes key: `CreateAnnotation`

Creates a new annotations XML file using the provided data.

```
POST http://localhost:3000/Document/q/Art/q?
DocumentID=u1234&AnnotationID=uanotherId
Content-Type: application/xml
<?xml version="1.0">...
```

## Successful Response:

200 OK

## Search Tasks

### Search Tasks

The *search task* API is designed for a viewer to perform server-side searching and text retrieval against the source document of a viewing session. A search task represents an asynchronous full-text search of a document and yields results as they become available.

### 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:

- `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 errorCode	Description
404		No viewing session with the provided <code>{viewingSessionId}</code> could be found.
480	<code>"DocumentNotProvidedYet"</code>	The viewing session does not yet have a source document attached.
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>"MissingInputForSimpleTerm"</code>	An invalid input value was used in a <code>"simple"</code> term object. See <code>errorDetails</code> in the response body.
480	<code>"InvalidInputForSimpleTerm"</code>	An invalid input value was used in a <code>"simple"</code> term object. See <code>errorDetails</code> in the response body.
480	<code>"MissingInputForProximityTerm"</code>	An invalid input value was used in a <code>"proximity"</code> term object. See <code>errorDetails</code> in the response body.

Code	Message	Description
480	"InvalidInputForProximityTerm"	An invalid input value was used in a "proximity" term object. See errorDetails 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	"InternalServerError"	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
/v2/viewingSessions/DLbVh9sTmXJAmD1GeXbS9Gn3WHxs8oib2xPsw2xEFjnIDdoJcudPtXciodSYFQq6zYGabQ_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

### 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
/v2/viewingSessions/DLbVh9sTmXJAmD1GeXbS9Gn3WHxs8oib2xPsw2xEFjnIDdoJcudPtXciodSYFQq6zYGabQ_rJIecdbkImTTkSA/searchTasks

Content-Type: application/json
```

```
"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
/v2/viewingSessions/DLbVh9sTmXJAmD1GeXbS9Gn3WHxs8oib2xPsW2xEFjnIDdoJcudPtxciodSYFQq6zYGabQ_rJIecdbkImTTkSA/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
/v2/viewingSessions/DLbVh9sTmXJAmD1GeXbS9Gn3WHxs8oib2xPsW2xEFjnIDdoJcudPtxciodSYFQq6zYGabQ_rJIecdbkImTTkSA/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
/v2/viewingSessions/DLbVh9sTmXJAmD1GeXbS9Gn3WHxs8oib2xPsW2xEFjnIDdoJcudPtxciodSYFQq6zYGabQ_rJIecdbkImTTkSA/searchTasks
```

```
{
  "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 processId 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 affinityToken of the search task. <b>Required when server clustering is enabled.</b>

### 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.
  - **boundingBox** (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.

- 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 errorCode	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:

```


```

```
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": "Cx07GHlkmi32gxAQhv49WZ",
  "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 /v2/searchTasks/pr5X6nPDgMwat6cx1mn0Q3/results?continueToken=Cx07GHlkmi32gxAQhv49WZ
Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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 /v2/searchTasks/pR5X6nPDgMwat6cx1mn0Q3/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]
}
```

(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
{processId}	The processId which identifies the search task.

### Request Headers

Name	Description
Accusoft-Affinity-Token	the affinityToken of the search task. <b>Required when server clustering is enabled.</b>

### Successful Response

HTTP/1.1 204 No Content

### Errored 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.
500	"InternalError"	The server encountered an internal error when handling the request.

## Viewing Package Creators

### Viewing Package Creators

The Viewing Package Creator API provides a facility to fully convert a document for viewing prior to the creation of a Viewing Session. If a large document is converted for viewing at the time a Viewing Session is created, the user may experience a delay when viewing later pages in the document. By converting a document prior to viewing, the entire document is available for viewing immediately.

*Note that these routes do not have route keys, as they cannot be re-routed.*

### POST /v2/viewingPackageCreators

Headers:

- Accusoft-Secret: mysecretkey

Body: a JSON object

The table below shows all the properties available for the POST Request:

input.source	{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 back-end services.
input.source.documentId	{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 <a href="#">RFC4648 - Base 64 Encoding with URL and Filename Safe Alphabet</a> .
input.source.type	{string}	Yes	Specify where PrizmApplicationServices can find the document. The valid values are "document", "url", or "upload".
input.source.fileName	{string}	Yes (If source.type="document")	Filename that the Storage Provider API can use.
input.source.url	{string}	Yes (If source.type="url")	URL to create a viewing session from.
input.source.headers	{object}	Optional (Used if source.type="url")	Request headers to send from PrizmApplicationServices when retrieving the URL.
input.source.acceptBadSslCertificate	{boolean=false}	Optional (Used if source.type="url")	If true, requires SSL certificates be valid. <i>Note: to use your own certificate authority, you need to specify an agent that was created with that CA as an option.</i>
input.source.displayName	{string}	Yes (if source.type="upload")	The display name of the document that's being

			uploaded. E.g. "Sample-2015-09-30T19:17:24Z.doc"
input.source.markupId	{string}	Optional	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.
input.source.downloadName	{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.
input.source.fileExtension	{string}	Optional	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" or "fileName" if this option is not provided.
input.viewingPackageLifetime	{integer}	Optional	Defines the minimum number of integer seconds for the created content to remain available. By default this is 24 hours. If set to 0, content will remain available perpetually.
input.render	{object}	Optional	The object that describes rendering options.
input.render.html5	{object}	Optional	The object that describes rendering options for the HTML5 viewer. A whitelist of the following render properties is used: <code>alwaysUseRaster</code> , <code>rasterResolution</code> , <code>svgMaxImageSize</code> and <code>vectorTolerance</code> . They are described in the

			PrizmDoc Server Viewing Session documentation.
minSecondsAvailable	{integer}	Optional (if a value is not provided, the default value = 3600 seconds will be used)	Defines the minimum number of seconds for the conversion process to remain available to GET status. By default this is 3600 seconds because by default, the Viewing Sessions timeout after 20m.

## Examples

### Create a ViewingPackage using a local document

POST http://localhost:3000/v2/viewingPackageCreators

Content-Type: application/json

```
{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.doc",
      "documentId": "unT67FxeKm81k1p0kPnyg8",
      . . .
    },
    "viewingPackageLifetime": 2592000
  }
}
```

### Successful Response:

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",
}
```

The response, along with other information, includes a new processId which is used to GET information about the converted content.

### **Errored 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.

```
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.

```
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*

```
480 MissingInput
Content-Type: application/json
{
  "errorCode": "MissingInput",
  "errorDetails": {
    "in": "body",
    "at": "input"
  }
}
```

```
480 InvalidInput
Content-Type: application/json
{
  "errorCode": "InvalidInput",
  "errorDetails": {
```

```
        at : input ,
        "expected": {
            "type": "object"
        }
    }
}
```

If a POST request is made with either a missing or an invalid header `Accusoft-Secret`, an error similar to the one below will be returned.

```
403 InvalidSecret
Content-Type: application/json
{
    "errorCode": "InvalidSecret"
}
```

### Create a ViewingPackage 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 http://localhost:3000/v2/viewingPackageCreators
Content-Type: application/json
{
    "input": {
        "source": {
            "type": "document",
            "fileName": "sample.doc",
            "documentId": "unT67FxeKm81k1p0kPnyg8",
            . . .
        },
        "render": {
            "html5": {
                "alwaysUseRaster": true
            }
        },
        "viewingPackageLifetime": 2592000
    }
}
```

### Successful Response:

```
200 OK
Content-Type: application/json
```

```
"input": {
  "source": {
    "type": "document",
    "fileName": "sample.doc",
    "documentId": "unT67FxeKm81k1p0kPnyg8",
    . . .
  },
  "render": {
    "html5": {
      "alwaysUseRaster": true
    }
  },
  "viewingPackageLifetime": 2592000
},
"expirationDateTime": "2015-12-09T06:22:18.624Z",
"processId": "khjyrfKLj2g6gv8fdqg710",
"state": "processing",
"percentComplete": 0
}
```

## Errored Responses:

### *Watermarks Input:*

The `input.watermarks` property cannot be used when creating a viewing package.

480 ReservedInput

Content-Type: application/json

```
{
  "errorCode": "ReservedInput",
  "errorDetails": {
    "in": "body",
    "at": "input.watermarks"
  }
}
```

## PUT /v2/viewingPackageCreators/{processId}/SourceFile

### *Headers:*

- Accusoft-Secret: mysecretkey

*Body:* a source document of Content-Type application/octet-stream

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

### Examples

PUT

http://localhost:3000/v2/viewingPackageCreators/khjyrfKLj2g6gv8fdqg710/SourceFile

Content-Type: application/octet-stream

Body: a document

### Successful Response:

200 OK

### Errored 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.

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.

580 SourceFileAlreadyProvided

Content-Type: application/json

```
{
  "errorCode": "SourceFileAlreadyProvided"
}
```

If a PUT request is made with either a missing or an invalid header `Accusoft-Secret`, an error similar to the one below will be returned.

403 InvalidSecret

Content-Type: application/json

```
{
  "errorCode": "InvalidSecret"
}
```

GET /v2/viewingPackageCreators/{processId}

- Accusoft-Secret: mysecretkey

Gets the status of an existing viewing package creator.

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".

### Examples

```
GET http://localhost:3000/v2/viewingPackageCreators/khjyrfKLj2g6gv8fdqg710
```

### Successful Response:

A response for a package that was successfully generated would look like this:

```
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:

- **Document Error**

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.

output property will be provided when the process completes in an error state.

```
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"
}
```

- **Password Errors**

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.

```
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,
}
```

},

- **Page Failures**

These errors will not mark the entire package as errored, but rather report in the output object in a `pageFailures` array.

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",
    "pageFailures": [
      {
        "errorCode": "CouldNotProvideRasterContent",
        "pageNumber": "4"
      },
      {
        "errorCode": "CouldNotProvideSvgContent",
        "pageNumber": "5"
      },
      {
        "errorCode": "CouldNotProvideTextContent",
        "pageNumber": "8"
      }
    ]
  },
  "expirationDateTime": "2015-12-09T06:22:18.624Z",
  "processId": "khjyrfKLj2g6gv8fdqg710",
  "state": "complete",
  "percentComplete": 100
}
```

- **Attachment Errors**

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.

Content-Type: application/json

```
{
  "input": {
    "source": {
      "type": "document",
      "fileName": "sample.doc",
      "documentId": "unT67FxeKm8lk1p0kPnyg8",
      . . .
    },
    "viewingPackageLifetime": 2592000
  },
  "output": {
    "packageExpirationDateTime": "2016-1-09T06:22:18.624Z",
    "attachments": [
      {
        "name": "letter.doc",
        "pageFailures": [
          {
            "errorCode" : "CouldNotProvideRasterContent",
            "pageNumber" : "2"
          },
          {
            "errorCode" : "CouldNotProvideSvgContent",
            "pageNumber" : "4"
          }
        ]
      },
      {
        "name": "re.eml",
        "attachments": [
          {
            "name": "budget1997.xls"
          }
        ]
      },
      {
        "name": "some-exe",
        "errorCode": "UnrecognizedFileFormat"
      }
    ]
  },
  "expirationDateTime": "2015-12-09T06:22:18.624Z",
  "processId": "khjyrfKLj2g6gv8fdqg710",
  "state": "complete",
  "percentComplete": 100
}
```

## Errored Response:

If a GET request is made with either a missing or an invalid header `Accusoft-Secret`, an error similar to the one below will be returned.

```
403 InvalidSecret
Content-Type: application/json
{
  "errorCode": "InvalidSecret"
}
```

## Viewing Packages

### Viewing Packages

*Note that these routes do not have route keys, as they cannot be re-routed.*

GET `/v2/viewingPackages/{documentId}`

Headers:

- `Accusoft-Secret: mysecretkey`

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 `http://localhost:3000/v2/viewingPackages/unT67FxeKm81k1p0kPnyg8`

## Successful Response:

```
200 OK
Content-Type: application/json
{
  "source": {
    "type": "document",
    "fileName": "sample.doc",
    "documentId": "unT67FxeKm81k1p0kPnyg8"
  },
  "state": "complete",
  "packageExpirationDateTime": "2016-1-09T06:22:18.624Z",
  "pageFailures": [
```

```
        "errorCode": "CouldNotProvideRasterContent",
        "pageNumber": "4"
    },
    {
        "errorCode": "CouldNotProvideSvgContent",
        "pageNumber": "5"
    }
]
}
```

## Errored Response:

If a GET request is made with either a missing or an invalid header `Accusoft-Secret`, an error similar to the one below will be returned.

```
403 InvalidSecret
Content-Type: application/json
{
    "errorCode": "InvalidSecret"
}
```

## GET /v2/viewingPackages/{documentId}/creator

Headers:

- `Accusoft-Secret: mysecretkey`

This URL can provide a viewing package's creator process ID. A viewing package may outlive the process that created it, in which case a 404 response should be expected. When pre-converting viewing packages, the creator process ID is provided in response to the initial POST request. This URL is primarily useful to access to the creator process when [creating a viewing session](#) with caching enabled (when a `documentId` is provided) in which case the viewing package creator process is internally created by PrizmDoc Application Services.

## Example

```
GET http://localhost:3000/v2/viewingPackages/unT67FxeKm81k1p0kPnyg8/creator
```

## Successful Response:

```
200 OK
Content-Type: application/json
{
    "viewingPackageCreatorId": "khjyrfKLj2g6gv8fdqg710"
}
```

## Errored Responses:

returned.

404 Not Found

If a GET request is made with either a missing or an invalid header `Accusoft-Secret`, an error similar to the one below will be returned.

```
403 InvalidSecret
Content-Type: application/json
{
  "errorCode": "InvalidSecret"
}
```

## DELETE /v2/viewingPackages/{documentId}

*Headers:*

- `Accusoft-Secret: mysecretkey`

When the viewing package is no longer needed, a DELETE request to this URL will remove all content and delete the resource.

### Example

```
DELETE http://localhost:3000/v2/viewingPackages/unT67FxeKm81k1p0kPnyg8
```

### Successful Response:

If a DELETE request is made the content will be marked for asynchronous deletion.

204 No Content

### Errored Responses:

If a DELETE request is made with an invalid package ID, an error will be returned.

404 Not Found

If a DELETE request is made with either a missing or an invalid header `Accusoft-Secret`, an error similar to the one below will be returned.

```
403 InvalidSecret
Content-Type: application/json
{
  "errorCode": "InvalidSecret"
}
```

## viewing session

### Viewing Session

#### POST /ViewingSession

##### Routes key: `PostViewingSession`

Body: a JSON object

PrizmApplicationServices will handle properties in the source object and any additional **ViewingSession properties** defined outside of the source object will be forwarded to the PrizmDoc Server when creating the Viewing Session. Please use the below watermarks example as a reference.

Property	Type	Description
source	{object}	Properties relevant to PrizmApplicationServices will reside in this object. <i>Note: This object will be removed before sending the rest of the body to PCCIS.</i>
source.type	{string}	Specify where PrizmApplicationServices can find the document. Can be "document", "url", "upload", or "viewingPackage".
source.fileName	{string}	(Required if source.type="document"). Filename that the Storage Provider API can use.
source.url	{string}	(Required if source.type="url"). URL to create a viewing session from.
source.headers	{object} (optional)	(Optionally used if source.type="url"). Request headers to send from PrizmApplicationServices when retrieving the URL.
source.acceptBadSslCertificate	{boolean=false} (optional)	(Optionally used if source.type="url"). If true, requires SSL certificates be valid. <i>Note: to use your own certificate authority, you need to specify an agent that was created with that CA as an option.</i>
source.displayName	{string}	(Required if source.type="upload"). The display name of the document that's being uploaded. E.g. "Sample-2015-09-30T19:17:24Z.doc"
source.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.
source.downloadName	{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.
source.fileExtension	{string} (optional)	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" or "fileName" if this option is not provided.

source.documentId	{string}	(Required if source.type="viewingPackage", otherwise optional). 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 <a href="#">RFC4648 - Base 64 Encoding with URL and Filename Safe Alphabet</a> . If source.type="viewingPackage", this value must refer to a complete viewing package, otherwise the viewing session will not be created and an error will be returned. For all other source types this value can optionally be provided. If the value refers to a complete viewing package it will be used for the viewing session. If a viewing package does not exist for the given documentId, the current request will be forwarded to PCCIS and a new viewing package creator process will be started which, once complete, can be used by viewing sessions using the same documentId value. If the value refers to a viewing package in error, the request will be forwarded to PCCIS and the viewing package must expire or be deleted before another can be created.
-------------------	----------	--

## Examples

### Create a session using a local document

Use Storage Provider to find the correct document and uses it to create a viewing session.

```
POST http://localhost:3000/ViewingSession
Content-Type: application/json
{
  "source": {
    "type": "document",
    "fileName": "sample.doc"
  }
}
```

### Successful Response:

```
200 OK
Content-Type: application/json
{ "viewingSessionId": "{viewingSessionId}" }
```

### Errored Response:

```
Content-Type: application/json
{ "errorCode": "DocumentNotFound" }
```

## Create a session using a local document with caching enabled

Use Storage Provider to find the correct document and use it to create a viewing session. Asynchronously, a viewing package creator process will begin creating a viewing package for the same document which, once complete, can be used by viewing sessions created from an identical request or by viewing sessions created with `source.type="viewingPackage"`.

*Note: When caching is enabled, watermarks cannot be used and will return an error.*

```
POST http://localhost:3000/ViewingSession
Content-Type: application/json
{
  "source": {
    "type": "document",
    "fileName": "sample.doc"
    "documentId": "doc_9495837910qc"
  }
}
```

## Successful Response:

```
200 OK
Content-Type: application/json
{ "viewingSessionId": "{viewingSessionId}" }
```

## Errored Responses:

*Watermarks Input:*

The `watermarks` property cannot be used when creating a viewing session with caching.

```
480
Content-Type: application/json
{
  "errorCode": "InputNotSupportedWithDocumentId",
  "in": "body",
  "at": "watermarks"
}
```

*render.html5.alwaysUseRaster Input:*

The `render.html5.alwaysUseRaster` property cannot be used when creating a viewing session with

```
480
Content-Type: application/json
{
  "errorCode": "InputNotSupportedWithDocumentId",
  "in": "body",
  "at": "render.html5.alwaysUseRaster"
}
```

### *Conflicting Input:*

When creating a viewing session from a completed viewing package, the `source` and `render` properties must match those from the package. When attempting to create a viewing session from a package with different `render` or `source` properties, PAS will return an error similar to this one:

```
480
Content-Type: application/json
{
  "errorCode": "InputConflictsWithViewingPackage",
  "in": "body",
  "at": "source.fileName"
}
```

### **Create a session using a url**

```
POST http://localhost:3000/ViewingSession
Content-Type: application/json
{
  "source": {
    "type": "url",
    "url": "http://google.com/",
    "headers": {
      "user-agent": "PrizmShare"
    },
    "acceptBadSslCertificate": true
  }
}
```

### **Successful Response:**

```
200 OK
Content-Type: application/json
{ "viewingSessionId": "{viewingSessionId}" }
```

## Create a session using a viewing package

Use a complete viewing package to start a viewing session. All viewable content for the session is immediately available by the viewing package. See the [PrizmDoc Application Services RESTful Viewing Package Creators API](#) for more details about creating viewing packages and the [PrizmDoc Application Services RESTful Viewing Packages API](#) for more details about managing viewing packages.

```
POST http://localhost:3000/ViewingSession
Content-Type: application/json
{
  "source": {
    "type": "viewingPackage",
    "documentId": "doc_9495837910qc"
  }
}
```

## Successful Response:

```
200 OK
Content-Type: application/json
{ "viewingSessionId": "{viewingSessionId}" }
```

## Errored Response:

```
480 InvalidInput
Content-Type: application/json
{
  "errorCode": "InvalidInput",
  "in": "body",
  "at": "source.documentId",
  "expected": {
    "type": "string"
  }
}
```

## Passing Additional parameters to the back end

```
POST http://localhost:3000/ViewingSession
Content-Type: application/json
{
  "source": {
    "type": "document",
    "fileName": "sample.doc"
  },
}
```

```
{
  "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"
}
]
```

## Successful Response:

```
200 OK
Content-Type: application/json
{ "viewingSessionId": "{viewingSessionId}" }
```

## Create a session by uploading a document

Creates a new blank session

```
POST http://localhost:3000/ViewingSession
Content-Type: application/json
{
  "source": {
    "type": "upload",
    "displayName": "sample_2015-10-31T19:15:32Z.doc"
  }
}
```

While the above is the minimum required data, when uploading a file, it is also a good idea to add the `markupId` property in `source`, as such:

```
{
  "source": {
    "type": "upload",
    "displayName": "sample_2015-10-31T19:15:32Z.doc",
    "markupId": "YSB1bm1xdWUgdmFsdWU="
  }
}
```

When this value is provided, PAS will use it when saving and reading markup files for this document. This ID needs to be unique for the binary document being used. Note that if your system contains mutable documents -- one that the user can edit and the system will still see it as a separate document -- this ID needs to take into account the revision of the document, as an edited version of the same document is considered a new, unique document by PrizmDoc. If this ID is not provided, PAS will hash the `displayName` value, which would in turn have the same uniqueness requirements.

## Successful Response:

```
200 OK
Content-Type: application/json
{ "viewingSessionId": "{viewingSessionId}" }
```

## PUT /ViewingSession/u{viewingSessionId}/SourceFile

### Routes key: `PutViewingSessionSourceFile`

Uploads a document to the session.

Headers:

- `Accusoft-Secret: mysecretkey`

Body: a document

### Examples

```
PUT http://localhost:3000/ViewingSession/u{viewingSessionId}/SourceFile
Accusoft-Secret: mysecretkey
Body: a document
```

## Successful Response:

```
200 OK
```

## Errored Response:

```
403 Forbidden
Content-Type: application/json
{ "errorCode": "InvalidSecret" }
```

## Developer Reference

This section contains the following information:

- PrizmDoc Server RESTful API
  - [Developer Reference](#)
  - [Attachments](#)
  - [Content Conversion Service](#)
  - [Form Extractors](#)
  - [Health Status](#)
  - [HTML5 Viewing](#)
  - [License Viewer \(Deprecated\)](#)
  - [Markup Burners](#)
  - [Multi-Server Mode](#)
  - [Redaction Creator](#)
  - [Search Contexts](#)
  - [Search Tasks](#)
  - [System Information \(Deprecated\)](#)
  - [Viewing Sessions](#)
  - [Work Files](#)

## Attachments

Viewing EML and MSG attachments using PrizmDoc Server.

### GET ViewingSession/u{ViewingSessionID}/Attachments

This RESTful API returns a JSON List object Attachments containing the collection 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 denotes whether the list is completed or not thus whether the request must be retried again. Each Attachment object in the list contains **displayName** and **viewingSessionId** property. The **displayName** can be used to label a href link on a web page and the link points to the **viewingSessionId** URL for the attachment. Please see the PrizmDoc Server [samples](#) for more details.

#### Http Method

GET

#### Resource URL

/PCCIS/V1/ViewingSession/u{ViewingSessionID}/Attachments

#### Parameters

ViewingSessionID	The ID of the viewing session associated with the request.
------------------	--

#### Request Body

## Response Body

If successful, this method returns the following properties:

Property Name	Value	Description
status	String	Specifies the current status of the attachments. <ul style="list-style-type: none"><li>"pending" means that there may be attachments, but the list has not yet been constructed.</li><li>"complete" means that the list is known and present in this object.</li></ul>
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.

## Example

```
GET http://localhost:18681/PCCIS/V1/ViewingSession/uGcIsIsEGbLV2_V9yy4NzmK2HB-JuLOH--A9sZl6cla9tx00ZDBGfqlG4kKu0r_GyEps4wWCvDwn4dpnZAR76Uw/Attachments
```

## Example Response

```
HTTP 200
Content-Type: application/json
{
  "attachments": [
    {
      "displayName": "example-file.pdf",
      "viewingSessionId": "SC0fZEMyIiGdOPMKBqLY8P6EAnVLEqxeTHjUUyqqxgJJC3s3wsQ8Lw2qqvkD1uLKTpAlF1ce23EQ6BFpYb4E3LC9TsXxwHCgB-I1c5rPOt0"
    },
    {
      "displayName": "second-example.doc",
      "viewingSessionId": "33qQovKTjc0UKbNgOI-5POEyCpNw5x-uEzGMB13iUVhnCa_UHSSnOpRBEzPKed7Maxq2RQu2S0OwJj14X4iU_650Qjx2EI5-7h-bXlYc6uA"
    }
  ],
  "status": "complete"
}
```

## Content Conversion Service

### Content Conversion

The Content Conversion API allows you to convert files from a [variety of input formats](#) to [several common output formats](#).

To convert a file:

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	Purpose
<a href="#">POST /v2/contentConverters</a>	Create and start a content conversion
<a href="#">GET /v2/contentConverters/{processId}</a>	Get the status of a content conversion

Note that these URLs begin with /v2, **not** /PCCIS/V1.

### POST /v2/contentConverters

status and final results of the conversion.

## Request Headers

Name	Value	Details
Content-Type	application/json	required
Accusoft-Affinity-Token	Affinity token returned in post response body for work file specified by <code>fileId</code> parameter in request body. Example: "rcqmuB9pAa8+4V7fhO1SXzawy/YMQU1g8LdNdE5l7w="	Only required if PrizmDoc is running in <a href="#">multi-server mode</a> .

## 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 http://localhost:18681/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>	The id of the <a href="#">WorkFile</a> to use as input. See <a href="#">Supported Input File Formats</a>	string, <b>required</b> 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>	The password to be used for a document associated with the <code>fileId</code> . Currently <code>password</code> is only supported when the source format is PDF, MS Word, MS Excel, MS PowerPoint or OpenDocument, and is ignored otherwise. Please note that only Office Open XML versions of MS Word, MS Excel and MS PowerPoint 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"

### 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>	The id of the <a href="#">WorkFile</a> to use as input. See <a href="#">Supported Input File Formats</a>	string, <b>required</b> Example: "ek5Zb123oYHSUEVx1bUrVQ"

### input.dest

The `input.dest` object specifies the destination file format and any additional details which control how the content is converted.

<code>input.dest.format</code>	Specifies the output file format. Must be one of the following: <ul style="list-style-type: none"> <li>"jpeg"</li> <li>"pdf"</li> <li>"png"</li> <li>"svg"</li> <li>"tiff"</li> </ul>	string, <b>required</b>
<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>	Specifies the header to be appended to each page of a document. The original page content will be left unaltered. The overall page dimensions will be expanded to allow space for the additional header content.	object, optional
<code>input.dest.footer</code>	Specifies the footer to be appended to each page of a document. The original page content will be left unaltered. The overall page dimensions will be expanded to allow space for the additional footer content.	object, optional

#### input.dest.jpegOptions

Name	Description	Details
<code>input.dest.jpegOptions.maxWidth</code>	The maximum pixel width of the output image, expressed as a CSS-style string, e.g. "800px". When specified, the output image is guaranteed to never be wider than the specified value and its aspect ratio will be preserved. This is useful if you need all of your output images to fit within a single column.	string, optional Example: "800px"
<code>input.dest.jpegOptions.maxHeight</code>	The maximum pixel height of the output image, expressed as a CSS-style string, e.g. "600px". When specified, the output image is guaranteed to never be taller than the specified value and its aspect ratio will be preserved. This is useful if you need all of your output images to fit within a single row.	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>	If <code>true</code> , 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 <code>false</code> .	boolean, optional

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.

#### input.dest.pngOptions

Name	Description	Details
<code>input.dest.pngOptions.maxWidth</code>	The maximum pixel width of the output image, expressed as a CSS-style string, e.g. "800px". When specified, the output image is guaranteed to never be wider than the specified value and its aspect ratio will be preserved. This is useful if you need all of your output images to fit within a single column.	string, optional Example: "800px"
<code>input.dest.pngOptions.maxHeight</code>	The maximum pixel height of the output image, expressed as a CSS-style string, e.g. "600px". When specified, the output image is guaranteed to never be taller than the specified value and its aspect ratio will be preserved. This is useful if you need all of your output images to fit within a single row.	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>	If <code>true</code> , the conversion process will produce single-page TIFF files, one file for each page of content (instead of a single TIFF with multiple pages). Default is <code>false</code> .	boolean, optional
<code>input.dest.tiffOptions.maxWidth</code>	The maximum pixel width of the output image, expressed as a CSS-style string, e.g. "800px". When specified, the output image is guaranteed to never be wider than the specified value and its aspect ratio will be preserved. This is useful if you need all of your output images to fit within a single column.	string, optional Example:

<code>input.dest.tiffOptions.maxHeight</code>	The maximum pixel height of the output image, expressed as a CSS-style string, e.g. "600px". When specified, the output image is guaranteed to never be taller than the specified value and its aspect ratio will be preserved. This is useful if you need all of your output images to fit within a single row.	string, optional Example: "600px"
---	--	--------------------------------------

For CAD input, you must specify either `maxWidth` or `maxHeight`.

## input.dest.header

Name	Description	Details
<code>input.dest.header.lines</code>	This is a multi-dimensional array that allows you to easily position text in a particular line and column. 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>	Specifies the name of the font that is used for the header (e.g. "fontFamily": "Courier"). The font name provided must be present on the server to be applied.	string, optional
<code>input.dest.header.fontSize</code>	Specifies the size of the font, in points. If provided, the value must be a string with a number followed by "pt" (e.g. "12pt").	string, optional
<code>input.dest.header.color</code>	Specifies the color of the text. Valid values are any valid CSS HEX value (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>	This is a multi-dimensional array that allows you to easily position text in a particular line and column. 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>	Specifies the name of the font that is used for the footer (e.g. "fontFamily": "Courier"). The font name provided must be present on the server to be applied.	string, optional
<code>input.dest.footer.fontSize</code>	Specifies the size of the font, in points. If provided, the value must be a string with a number followed by "pt" (e.g. "12pt").	string, optional
<code>input.dest.footer.color</code>	Specifies the color of the text. Valid values are any valid CSS HEX value (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](#).

## 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 value is 60, ensuring that you have at least 60 seconds to get the result status of any conversion operation.

## Response Body

A successful response will return JSON which contains:

1. The `input` object submitted in the request, normalized to include default values.
2. Information about the `status` of the conversion.

Here is an example:

```
200 OK
Content-Type: application/json
{
  "input": {
    "sources": [
      {
        "fileId": "ek52b123oYHSUEVx1bUrVQ",
        "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"> <li>"processing" - The conversion is still in progress.</li> <li>"complete" - The conversion has completed successfully.</li> <li>"error" - The conversion failed due to a problem.</li> </ul> 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 is running in <a href="#">multi-server mode</a> .	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	N/A	The contentConverter was created and the conversion process was started.
400	error	CouldNotReadRequestData	Could not read request data.
405	N/A	N/A	POST HTTP method was not used.
480	error	InvalidJson	Json error details are in <a href="#">errorDetails</a> .
480	N/A	InvalidDimensionValue	Invalid dimension value specified for rasterization. See details in <a href="#">errorDetails</a> .
480	N/A	InvalidInput	Invalid input. Invalid request data is referenced in the <a href="#">errorDetails</a> .
480	N/A	InvalidPageSyntax	Invalid page specification. See <a href="#">errorDetails</a> .
480	N/A	ForceOneFilePerPageNotSupportedWhenUsingHeaderOrFooter	forceOneFilePerPage mode is not supported when using header or footer options. Supported forceOneFilePerPage option is referenced in <a href="#">errorDetails</a> .
480	N/A	MaxWidthOrMaxHeightMustBeSpecifiedWhenRasterizingCadInput	Max width or max height must be specified when rasterizing CAD input. See <a href="#">errorDetails</a> .
480	N/A	MissingInput	Missing input. See <a href="#">errorDetails</a> .
480	N/A	MultipleSourcesAreNotSupportedForThisDestinationFormat	Multiple source files or pages are not supported for this destination format.
480	N/A	MultipleSourceDocumentsNotSupportedWhenUsingHeaderOrFooter	Multiple source documents are not supported when using header or footer.
480	N/A	PagesPropertyNotSupportedWhenUsingHeaderOrFooter	Pages property is not supported for conversion with header or footer. The property is referenced in <a href="#">errorDetails</a> .
480	N/A	UnrecognizedExpression	Unrecognized expression. See <a href="#">errorDetails</a> .
480	N/A	UnsupportedConversion	Unsupported conversion. See <a href="#">errorDetails</a> .

status	body		
480	N/A	UnsupportedDestinationFormatWhenUsingHeaderOrFooter	Unsupported destination format when using header or footer. Supported destination formats are listed in <b>errorDetails</b> .
480	N/A	UnsupportedSourceFileFormat	Unsupported source file format. Unsupported file is referenced in <b>errorDetails</b> .
480	N/A	WorkFileDoesNotExist	Specified work file does not exist.
580	N/A	InternalServerError	Internal service error. This error can be returned for a number of different reasons. Please contact 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
processId	The processId for a particular contentConverter. This processId was returned in the response for the initial POST.	string, <b>required</b>

### Request Headers

Name	Value	Details
Accusoft-Affinity-Token	Affinity token returned in post response body for content converter specified by processId parameter. Example: "rcqmuB9pAa8+4V7fh01SXzawy/YMQU1g8lLdNDe5I7w="	Only used if PrizmDoc is running in <b>multi-server mode</b> .

### Response Body

While processing, the response will return JSON with only the processing details. For example:

```
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:

```
200 OK
Content-Type: application/json
{
  "input": {
    "sources": [
      {
```

```
    }
  ],
  "dest": {
    "format": "pdf",
    "pdfOptions": {
      "forceOneFilePerPage": false
    }
  }
},
"expirationDateTime": "2015-12-17T20:38:39.796Z",
"processId": "ElkNzWtrUJp4rXI5YnLUgw",
"state": "complete",
"percentComplete": 100,
"output": {
  "results": [
    {
      "fileId": "KOrSwaqsguevJ97BdmUbXi",
      "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:

```
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+JQPpy0POu+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

	<ul style="list-style-type: none"> <li>processing - The conversion is still in progress.</li> <li>complete - The conversion has completed successfully.</li> <li>error - The conversion failed due to a problem.</li> </ul>	
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 is running in <a href="#">multi-server mode</a> .	string Example: "rcqmuB9pAa8+4V7fhO1SXzawy/YMQU1g8LdNdE5l7w="

## Conversion Output Details

Name	Description	Details
output.results	An array of objects, one for each output file created.	object
output.results[n].fileId	The <a href="#">WorkFile</a> id for an output file. Use this id to download the output file using the <a href="#">WorkFile API</a> .	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 <a href="#">WorkFile</a> 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, 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".	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	N/A	N/A	N/A	The contentConverter was created and the conversion process was started.
200	complete	N/A	N/A	N/A	The conversion process was completed.
200	complete	N/A	output.results[n].errorCode	NoSuchPage	No such page. Problem field 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 field is listed in <code>output.results[n].sources[0].fileId</code>
200	error	CouldNotConvert	output.results[n].errorCode	CouldNotConvertPage	Could not convert page. Problem field 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 field, 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	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 field is listed in <code>output.results[n].sources[0].fileId</code> .  For CouldNotConvertPage, InvalidPassword and NoSuchPage errors, problem field and pageNumber are listed in <code>output.results[n].sources[0].fileId</code> .

status	JSON body	body	location in response JSON body	body	Description
					output.results[n].sources[0].page
404	N/A	ContentConverterDoesNotExist	N/A	N/A	Content converter does not exist. Invalid processId was specified in the request.
405	N/A	N/A	N/A	N/A	POST HTTP method was not used.
580	N/A	InternalServerError	N/A	N/A	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

## Form Extractors

### Form Extractors

The *form extractors* API allows you to detect form field elements in PDF and raster documents.

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 viewingSessionId 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:

- "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 errorCode	Description
404		No viewing session with the provided {viewingSessionId} could be found.
480	"DocumentNotProvidedYet"	A source document has not been provided to the viewing session.
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> . <b>Required when server clustering is enabled.</b>

#### 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.

### 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.

- `"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 errorCode	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/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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
Accusoft-Affinity-Token	The affinityToken of the form extraction process. <b>Required when server clustering is enabled.</b>

## 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 Accusoft-Affinity-Token request header was not provided.
404		No <i>form extractor</i> could be found for the given {processId}.
500	"InternalError"	The server encountered an internal error when handling the request.

## Example

### Request

```
GET /v2/formExtractors/gLo1tqCVnRKzXz2QFNptqw
Accusoft-Affinity-Token: D+Rmn9k84FrLfrHoNL2bag6WpuNn2ox2qhT2GbLdf9A=
```

### 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": [
```

```
    "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": "gL0ltqCVnRKzXz2QFNptqw",
"state": "complete",
"affinityToken": "D+Rmn9kB4FrLfrHoNL2bag6WpuNn2ox2qhT2GbLdf9A="
}
```

## "acroform" Output

The `output.acroform` object will conform to the following. All properties are always present unless otherwise noted:

- `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.
      - `maxLength` (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

- "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).

- **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.

## Health Status

### GET /Service/Current/Health

Returns the following HTTP status codes to reflect the overall health of the PrizmDoc Server:

- 200 - If the system is running and no health issues exist.
- 500 - If the system some or all of the PrizmDoc Server are unhealthy.

This URL is intended to provide a quick, easily parseable indication as to the health of PrizmDoc Server. For more information about the current health, use GET /Service/Current/Info.

#### Http Method

GET

#### Resource URL

/PCCIS/V1/Service/Current/Health

#### Parameters

None

#### Request Body

None

#### Response Body

OK	for HTTP status code 200
None/Empty	for HTTP status code 500

GET <http://localhost:18681/PCCIS/V1/Service/Current/Health>

## Example Response

200 OK

## GET Service/Current/Info

This API returns a JSON object indicating the health of PrizmDoc Server.

### Http Method

GET

### Resource URL

/PCCIS/V1/Service/Current/Info

### Parameters

None

### Request Body

None

### Response Body

If successful, this method returns the following properties:

Property Name	Value	Description
serviceStatus	String	Status of PrizmDoc Server: <ul style="list-style-type: none"><li>• "starting"</li><li>• "running"</li><li>• "unhealthy"</li></ul>
licenseStatus	String	Information about the PrizmDoc license. <ul style="list-style-type: none"><li>• "not licensed"</li><li>• "licensed as '...'"</li></ul>
instances[].serviceStatus	String	Status of the PrizmDoc Server used for Viewing.
instances[].serviceInstallerVersion	String	Version of the installer used to install PrizmDoc Server.
instances[].pccisVersion	String	Version of the PrizmDoc Server used for Viewing.
instances[].runtimeVersion	String	.NET runtime version supporting the PrizmDoc Server used for Viewing.
instances[].operatingSystem	String	Operating System of the server on which PrizmDoc Server are running.
instances[].startTime	String	Last recorded time the PrizmDoc Server were started. Time is reported in UTC and is ISO-8601 format.
instances[].instanceId	String	Host name of the server running PrizmDoc Server.
instances[].childServices	Array	Name and status for each individual child PrizmDoc Server.

instances[].childServices[].status   String   Status of the child PrizmDoc Server:

- "starting"
- "running"
- "unhealthy"

### Example

```
GET http://localhost:18681/PCCIS/V1/Service/Current/Info
```

### Example Response

```
{
  "serviceStatus": "running",
  "licenseStatus": "licensed as 'PCC' ",
  "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",

```

```
,  
{  
  "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"  
}  
]
```

```
}
```

## HTML5 Viewing

HTML5 viewing information.

GET Page/q/{PageNumber}/Attributes?DocumentID=[e,u]{ViewingSessionId}

This API requests or reads the information about a particular document page. The return information is a JSON string. The form has a query parameter, **DocumentID**, which can have for its value an un-encoded "u" prefix followed by the viewing session ID or a base 64 encoded viewing session ID prefixed with "e". The **{PageNumber}** parameter always begins with zero (0) for the first page within a document.

### Http Method

GET

### Resource URL

/PCCIS/V1/Page/q/{pageNumber}/Attributes?DocumentID=[e,u]{ViewingSessionId}& ContentType=[png, svg, svgb]

### Parameters

PageNumber	The page number of the document being requested. The first page within a document is always 0.
DocumentID	The Viewing Session ID.
ContentType	(Optional). The intended content of the page image.

### Request Body

Empty.

### Response Body

If successful, this method returns an object with the following properties:

Property Name	Value	Description
version	String	The version of the Viewer to consume this request.
contentType	String	The type of content that is available for this document.
imageBitDepth	Integer	The bit depth of the viewable content.
imageHeight	Integer	The height of the viewable content.
imageWidth	Integer	The width of the viewable content.

imageYResolution	Integer	The Y resolution of the viewable content.
------------------	---------	---

## Example

```
GET http://localhost:18681/PCCIS/V1/Page/q/0/Attributes?DocumentID=eOQA...
```

## Example Response

```
{"version": "7.1", "contentType": "jpeg, png, svg", "imageBitDepth": 16, "imageHeight": 990, "imageWidth": 765, "imageXResolution": 0, "imageYResolution": 0}
```

GET Page/q/{PageNumber}?DocumentID=[e,u]{ViewingSessionId}&Scale={Scale}&ContentType=[jpeg, png, svg, svga, svgb]

This RESTful API returns a full page image. The **Scale** and **ContentType** query parameters affect the image with respect to the parameters intent. The **ContentType** parameter may have the following parameter value for now: JPEG, PNG, SVG, SVGA, or SVGB. If SVG, SVGA, or SVGB is chosen, the scale values are ignored as a SVG (vectored) stream is returned for which this parameter has no practical meaning. The scale query parameter with a value 1.0 gives a non-resized image. Less than 1.0 values shrink the image and larger than 1.0 values increase the image size.

 SVGA is standard SVG content with additional optimizations to reduce the payload size. SVGA is not compatible with the Internet Explorer 10 browser, in which case SVG should be used instead.

 SVGB is standard SVG content additionally optimized by replacing SVG curves representing font letters with Unicode font to reduce payload size. Note: "svgb" provides better optimization and visually more consistent results. SVGB is not compatible with the Internet Explorer 9 & 10 browser, desktop Safari 9.x or Chrome on Windows in which case SVG, or SVGA for IE 9, should be used instead.

The API will now interpret an optional request header:

### Accept-Encoding: gzip[,deflate,sdch]

The API will look for the value 'gzip' to mean that the Viewer is able to handle Gzipped SVG/SVGA/SVGB content. Based on this header value, the API will compress the response SVG/SVGA/SVGB content using the Gzip format. It will add the following header in the response:

### Content-Encoding: gzip

 Uncompressed data lengths less than 500 bytes will not be compressed.

Most modern browsers can handle Gzipped content. If you would like to see reduction in the network bandwidth usage for the SVG/SVGA/SVGB content in the response, then make sure that your webtier/proxy does not strip these headers.

 Raster Images (content types : png,jpg) will not be Gzip compressed.

## Http Method

GET

## Resource URL

[jpeg, png, svg, svga, svgb]

## Parameters

PageNumber	The page number of the document being requested. The first page within a document is always 0.
DocumentID	The Viewing Session ID.
Scale	The intended scale value of the page image. This is ignored for SVG streams.
ContentType	The intended content type of the page image.

## Request Header

(optional)

Accept-Encoding: gzip[,deflate,sdch]

## Request Body

Empty.

## Response Header

Content-Encoding: gzip

Note: This header will exist only if the request contained the optional header, Accept-Encoding: gzip.

## Response Body

If successful, this method returns either the requested page content or, if the requested svg content type cannot be provided, an object is returned containing an errorCode with the value SvgNotAvailable and the pageAttributes for png content type. This is provided as an optimization to fallback to png when svg is not available.

## Example

```
GET http://localhost:18681/PCCIS/V1/Page/q/0?DocumentID=eOQA5...  
&Scale=1&ContentType=svg
```

## Example Response

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?> <svg  
xmlns=http://www.w3.org/2000/svg xmlns:cc="http://creativecommons.org/ns#" ...  
</svg>
```

GET Page/q/{PageNumber}/Tile/{x}/{y}/{width}/{height}?DocumentID=[e,u]  
{ViewingSessionId}&Scale={Scale Value}&ContentType=png

This request returns a tile of the requested document page. The tile is formed from the pixel coordinate specified by the x and y parameters and the width and height parameters. This tile is resized as determined by the scale factor. The **contentType** should only be PNG for tiles as JPEG can cause artifact issues in alignment. Requesting SVG content will cause an error response as this content type is vector data and does

Less than 1.0 values shrink the tile and larger than 1.0 values increase the tile size.

## Http Method

GET

## Resource URL

/Page/q/{pageNumber}/Tile/{x}/{y}/{width}/{height}?DocumentID=[e,u]{ViewingSessionId}&Scale={Scale Value}&ContentType=png

## Parameters

PageNumber	The page number of the document being requested. The first page within a document is always 0.
DocumentID	The Viewing Session ID.
Scale	The intended scale value of the page image. This is ignored for SVG streams.
ContentType	The intended content type of the page image.
x	The x coordinate of the requested tile relative to the original image.
y	The y coordinate of the requested tile relative to the original image.
width	The width of the requested tile.
height	The height of the requested tile.

## Request Body

Empty.

## Response Body

If successful, this method provides the requested page tile content.

### Example

```
GET http://localhost:18681/PCCIS/V1/Page/q/0/ Tile/0/0/1275/1024?
DocumentID=eOAA5... &Scale=1&Quality=100&ContentType=png
```

### Example Response

Empty

GET Page/q/{PageNumber}/{Width}x{Height}?DocumentID=[e,u]  
{ViewingSessionId}&ContentType=[jpeg,png]

This request returns a thumbnail of the requested document page. The thumbnail size is constrained within the width and height pixel parameters and the aspect ratio of the image is maintained.

The **contentType** can only be PNG or JPEG.

## Http Method

## Resource URL

/Page/q/{pageNumber}/{width}x{height}?DocumentID=[e,u]{ViewingSessionId}&ContentType=[jpeg, png]

## Parameters

PageNumber	The page number of the document being requested. The first page within a document is always 0.
DocumentID	The Viewing Session ID.
ContentType	The intended content type of the thumbnail page image.
width	The maximum width of the requested thumbnail.
height	The maximum height of the height of the requested thumbnail.

## Request Body

Empty.

## Response Body

If successful, this method provides the requested page thumbnail content.

### Example

```
GET http://localhost:18681/PCCIS/V1/Page/q/0/ 200x300?DocumentID=eOAA5...
&ContentType=png
```

### Example Response

Empty

GET Document/q/Attributes?DocumentID=[e,u]  
{ViewingSessionId}&DesiredPageCountConfidence={PercentageValue}

This request returns a JSON string giving the page count of the document requested and the confidence value. Because page rendering may be different than what a file format may indicate for number of pages, there is a need for a confidence value indicating how confident the page count acquired will be prior to having all the pages generated. The **DesiredPageCountConfidence** parameter indicates what the minimum level required to be returned thus implying how long the request may take to return a precise value. The **DesiredPageCountConfidence** is segmented in this release where 50 percent or below may get an estimate page count value and above 50 percent will try to get a more accurate count. If a **pageCountConfidence** value of less than 100 percent is returned, it means that the request is still being worked on and to try again if an accurate count is still desired.

## Http Method

GET

## Resource URL

DocumentID	The Viewing Session ID.
DesiredPageCountConfidence	The intended percentage of confidence to use in determining the page count.

## Request Body

Empty.

## Response Body

If successful, this method returns an object with the following properties:

Property Name	Value	Description
pageCount	Integer	The determined number of pages in the document.
pageCountConfidence	Integer	If value is less than 100 it means that the page count is still being determined, and to try again if a more accurate count is needed.

## Example

```
GET http://localhost:18681/PCCIS/V1/ Document/q/Attributes?DocumentID=eOQA5...&DesiredPageCountConfidence=50
```

## Example Response

```
{"pageCount":3,"pageCountConfidence":100}
```

GET Document/q/{PageNumberBegin}-{PageNumberEnd}/Text?DocumentID=[e,u]{ViewingSessionId}

This request returns a JSON string containing the text extracted from the given document. The **{PageNumberBegin}** denotes the first page and **{PageNumberEnd}** denotes the last page to be included for text extraction. Additionally, information on hyperlinks will be returned if any is available. The lowest value for **{PageNumberBegin}** parameter is zero (0) – that indicates the first page within the document.

The API will now interpret an optional request header:

### Accept-Encoding: gzip[,deflate,sdch]

The API will look for the value 'gzip' to mean that the Viewer is able to handle Gzipped extracted text content. Based on this header information, the API will compress the extracted text data content using Gzip format. It will add the following header in the response:

### Content-Encoding: gzip

 Uncompressed data lengths less than 500 bytes will not be compressed.

Most modern browsers can handle Gzipped content. If you would like to see reduction in the network bandwidth usage for the extracted text content in the response, then make sure that your webtier/proxy does not strip these headers.

## Http Method

## Resource URL

/PCCIS/V1/ Document/q/0-2/Text?DocumentID=eMAA2...

## Parameters

DocumentID	The Viewing Session ID.
PageNumberBegin	The page to begin extracting text from.
PageNumberEnd	The page to end extracting text from.

## Request Header

(optional)

Accept-Encoding: gzip[,deflate,sdch]

## Request Body

Empty.

## Response Header

Content-Encoding: gzip

Note: This header will exist only if the request header contained the optional header, Accept-Encoding: gzip.

## Response Body

If successful, this method returns an object with the following properties:

Property Name	Value	Description
pages	List	A list of Nested Objects describing the extracted text for each of the pages in range.
pages.number	Integer	Denotes the number the current page to which the extracted text belongs.
pages.text	String	Extracted text of the current page.
pages.markup	List	A list of nested objects describing the hyperlinks if there are any.
pages.markup.changeType	String	The changeType will always be "Add".
pages.markup.markType	String	The markType will always be "DocumentHyperlink".
pages.markup.properties	Object	Object containing properties of the hyperlink.
pages.markup.properties.href	String	A value that indicates the destination of the Hyperlink.
pages.markup.properties.rectangle	Object	These values will be defined in points (1/72

pages.markup.properties.rectangle.x	Integer	The x location of the hyperlink markup to be applied.
pages.markup.properties.rectangle.y	Integer	The y location of the hyperlink markup to be applied.
pages.markup.properties.rectangle.width	Integer	The width of the hyperlink markup to be applied.
pages.markup.properties.rectangle.height	Integer	The height of the hyperlink markup to be applied.
pages.markup.properties.borderHorizontalRadius	Integer	The horizontal radius of the border of the markup to be applied.
pages.markup.properties.borderVerticalRadius	Integer	The vertical radius of the border of the markup to be applied.
pages.markup.properties.borderThickness	Integer	The width of the border of the markup to be applied.

## Example

```
GET http://localhost:18681/PCCIS/V1/Document/q/0-2/Text?DocumentID=eMAA2...
```

## Example Response - With Hyperlink Data

```
"pages":[{"number":0,"text":"Congratulations You have successfully installed Prizm Content Connect www.accusoft.com Page 1 of Document", "width":612, "height":792, "rectangles":[[97.8, 66.47, 43.32, 74.49], ..., [416.77, 520.44, 6.5, 32.28]], "markup":[{"changeType":"Add", "markType":"DocumentHyperlink", "properties":{"href":"http:www.accusoft.com", "rectangle":{"x":228.1, "y":329.0, "width":155.9, "height":23.0}, "borderHorizontalRadius":0.0, "borderVerticalRadius":0.0, "borderThickness":0.0}}]}]}
```

## Example Response - Without Hyperlink Data

```
{"pages":[{"number":0,"text":"Congratulations You have successfully installed Prizm Content Connect www accusoft com Page 1 of Document"}, {"number":1, "text":"Page 2 of Document"}, {"number":2, "text":"Page 3 of Document"}]}
```

## License Viewer (Deprecated)

 This API has been deprecated and will be removed from the public API in a future release.

## GET License/ClientViewer

There is a license URL which the client JavaScript viewer control (**pccViewer**) must use to enable viewing. The information sent by the control must be present in the Web Tier. The query parameters sent to this URL must be passed onto PrizmDoc Server. Information is returned back to the client control to enable viewing. If this communication sequence is not done or fails in any matter, no document views will be possible.

### Http Method

GET

### Resource URL

/PCCIS/V1/License/ClientViewer

### Parameters

v	License value 1.
iv	License value 2.
p	License value 3.

### Request Body

Empty.

### Response Body

If successful, this method returns a string used by the Viewer for licensing.

#### Example

```
GET http://localhost:18681/PCCIS/V1/License/ClientViewer
```

#### Example Response

```
K98u234982340/xdfwe/ Zsdfaslkj903941ke93asljfjsjf+ asfjoPopoijojiALKidije931kj0Y  
4aAq6FvBW7nqCSDfdefl1k3j3j=
```

## Markup Burners

### POST /MarkupBurner

Creates and starts a new MarkupBurner.

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 /MarkupBurner/{id} API 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 XML which defines the markup to burn, the other representing the source document on which to burn the markup. Refer to the [Work Files](#) API topic for more information.

# PrizmDoc v12.3 - Updated June 23, 2017

# 1106

document will be made available by a new WorkFile ID.

By default, MarkupBurner objects will be automatically deleted 20 minutes after they are created.

## Http Method

POST

## Request Headers

Name	Value	Details
Accusoft-Affinity-Token	Affinity token returned in post response bodies for work files specified by input.documentFileId and input.markupFileId parameter in request body.  Example: "rcqmuB9pAa8+4V7fhO1SXzawy/YMQU1g8lLdNDe5I7w="	Only required if PrizmDoc is running in <b>multi-server mode</b> .

## Parameters

None

## Request Body

In the request body, provide JSON containing the following properties:

Name	Description	Details
input.documentFileId	The ID of the WorkFile that represents the document to burn in the markup. This document will not be modified.	string, required  Example: ek5Zb123oYHSUEVx1bUrVQ
input.markupFileId	The ID of the WorkFile that represents the XML document which contains the markup definition.	string, required  Example: aQ1BdViqmUisBuevJKO9Sw
minSecondsAvailable	The minimum number of seconds which this MarkupBurner must remain available. If not provided, a configurable default value is used. This value is ignored if it is shorter than the configurable value.	integer, optional  Example: 60

## Response Body

If successful, this method returns JSON containing the following properties:

Name	Description	Details
input.documentFileId	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.	string  Example: ek5Zb123oYHSUEVx1bUrVQ
input.markupFileId	The ID of the WorkFile that represents the XML document which contains the markup definition. This is an echo of	string  Example: aQ1BdViqmUisBuevJKO9Sw

	request.	
expirationDateTime	The date and time (in ISO 8601 Extended Format) when the MarkupBurner will be deleted.	string Example: "2014-05-13T20:38:39.796Z"
processId	The ID of the MarkupBurner.	string Example: ElkNzWtrUJp4rXI5YnLUgw
state	The current state of the markup burning process running on the server.  This will always be <b>"processing"</b> in this response.	string ("processing"   "complete"   "error" ) Example: "processing"
percentComplete	The percentage (0 – 100) complete of the markup burning process.  This will always be <b>0</b> in this response.	integer Example: 0
errorCode	An error code string if a problem occurred during the markup burning process.  This will always be <b>null</b> in this response.	string Example: null
output.documentFileId	The ID of the new WorkFile that represents a new document with markup burned into it.  This will always be <b>null</b> in this response.	string
affinityToken	Affinity token echoed from request header. This value will only be present if PrizmDoc is running in multi-server mode.	string Example: "rcqmuB9pAa8+4V7fhO1SXzawy/YMQU1g8lLdNDe5I7w="

## Status Codes

- 200 OK, if MarkupBurner was created successfully.
- 400 Bad Request, if **input.documentFileId**, **input.markupFileId** is missing or invalid format, if **minSecondsAvailable** is not a number.
- 405 Method Not Allowed, if POST HTTP method is not used.

## Examples

### Example Request

```
POST http://localhost:18681/PCCIS/V1/MarkupBurner
Content-Type: application/json
{
```

```
        "markupFileId": "aQ1BdViqmUisBuevJKO9Sw"
    },
    "minSecondsAvailable": 60
}
```

## Example Response

```
200 OK
Content-Type: application/json
{
  "input": {
    "documentFileId": "ek5Zb123oYHSUEVx1bUrVQ",
    "markupFileId": "aQ1BdViqmUisBuevJKO9Sw"
  },
  "expirationDateTime": "2014-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "state": "processing",
  "percentComplete": 0,
  "errorCode": null,
  "output": null
}
```

## GET /MarkupBurner/{MarkupBurnerId}

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 Files](#) 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.

### Http Method

GET

### Request Headers

Name	Value	Details
Accusoft-Affinity-Token	Affinity token returned in post response body for markup burner specified by MarkupBurnerId parameter in URI. Example: "rcqmuB9pAa8+4V7fhO1SXzawy/YMQU1g8lLdNDe5I7w="	Only required if PrizmDoc is running in <a href="#">multi-server mode</a> .

### Parameters

Name	Description	Details
MarkupBurnerId	The ID of the MarkupBurner.	string, required Example: ElkNzWtrUJp4rXI5YnLUgw

## Request Body

None

## Response Body

If successful, this method returns JSON containing the following properties:

Name	Description	Details
input.documentFileId	The ID of the WorkFile that represents the document to burn in the markup. This document will not be modified.	string Example: ek5Zb123oYHSUEVx1bUrVQ
input.markupFileId	The ID of the WorkFile that represents the XML document which contains the markup definition.	string Example: aQ1BdViqmUisBuevJKO9Sw
expirationDateTime	The date and time (in ISO 8601 Extended Format) when the MarkupBurner will be deleted.	string Example: "2014-05-13T20:38:39.796Z"
processId	The ID of the MarkupBurner.	string Example: ElkNzWtrUJp4rXI5YnLUgw
state	The current state of the markup burning process running on the server.	string ("processing"   "complete"   "error" ) Example: "complete"
percentComplete	The percentage (0 – 100) complete of the markup burning process. <b>Note:</b> The percent complete will only ever be 0 or 100.	integer Example: 100
errorCode	An error code string if a problem occurred during the markup burning process.	string Example: "DocumentFileIdDoesNotExist"
output.documentFileId	The ID of the new WorkFile that represents a new document with markup burned into it.	string Example: vry3FPE0zQqYwhzndRccOQ
affinityToken	Affinity token echoed from request header. This value will only be present if PrizmDoc is running in multi-server mode.	string Example: "rcqmuB9pAa8+4V7fhO1SXzawy/YMQU1g8lLdNDe5I7w="

## Status Codes

- 200 OK, if MarkupBurner was retrieved successfully.
- 404 Not Found, if **MarkupBurnerId** does not exist.

## Error Codes

- **DocumentFileIdError** - This indicates that an error occurred attempting to access the document via the WorkFile Service. The specified workfile Id is valid, but an error prevented the file from being read.
- **DocumentFileIdDoesNotExist** - This error indicates that the document requested by the specified workfile Id does not exist.
- **MarkupFileIdError** - This indicates that an error occurred attempting to access the markup file via the WorkFile Service. In this case, the workfile Id is valid, but an error prevented the file from being read.
- **MarkupFileIdDoesNotExist** - This error indicates that the markup file requested by the specified workfile Id does not exist.
- **RedactionError** - This indicates a general error in creating a redacted document. Possible causes include failure to achieve an intermediate conversion, failure to delete an intermediate conversion, or failure to create the redacted output document.
- **RedactionWorkFileCreationError** - This indicates the markup burner failed to store the final redacted output file via the workfile service.

## Examples

### Example Request

```
GET http://localhost:18681/PCCIS/V1/MarkupBurner/ElkNzWtrUJp4rXI5YnLUgw
```

### Example Response

```
200 OK
Content-Type: application/json
{
  "input": {
    "documentFileId": "ek5Zb123oYHSUEVx1bUrVQ",
    "markupFileId": "aQ1BdViqmUisBuevJKO9Sw"
  },
  "expirationDateTime": "2014-12-17T20:38:39.796Z",
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "state": "complete",
  "percentComplete": 100,
  "errorCode": null,
  "output": {
    "documentFileId": "vry3FPE0zQqYwhzndRccOQ"
  },
}
```

## POST /PCCIS/V1/ViewingSession/u{ViewingSessionID}/MarkupBurner

Starts a new MarkupBurner using the source document of a viewing session and a provided XML markup definitions file 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.

GET /viewingsession/{viewingsessionid}/markupburner/{processid} URL below to know when the process has completed.

## HTTP Method

POST

## Parameters

None

## Request Body

The body of the request should be XML defining the markup that needs to be burned into a copy.

## Response Body

If successful, a JSON object which may contain:

Name	Description	Details
processId	The id of the process.	string Example: "ElkNzWtrUJp4rXI5YnLUgw"
state	The current state of the process. This will always be <b>"processing"</b> in the initial POST response.	string Example: "processing"
percentComplete	The percentage (0 - 100) complete of the process. This will always be <b>0</b> in the initial POST response.	integer Example: 0

## Status Codes

- 200 OK, if MarkupBurner was created successfully.
- 405 Method Not Allowed, if POST HTTP method is not used.

## Examples

### Example Request

```
POST https://localhost:18681/PCCIS/V1/ViewingSession/u{ViewingSessionID}/MarkupBurner
<<XML markup data>>
```

### Example Response

```
200 OK
Content-Type: application/json
{
  "processId": "ElkNzWtrUJp4rXI5YnLUgw",
  "state": "processing",
  "percentComplete": 0
}
```

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.

## HTTP Method

GET

## Parameters

None

## Request Body

None

## Response Body

If successful, a JSON object which may contain:

Name	Description	Details
processId	The id of the process.	string Example: "ElkNzWtrUJp4rXI5YnLUgw"
state	The current state of the process.	string ("processing"   "complete"   "error") Example: "complete"
percentComplete	The percentage (0 - 100) complete of the process.	integer Example: 100
errorCode	An error code string if a problem occurred during processing.	string

## Status Codes

- 200 OK, if the status of the MarkupBurner could be returned.
- 404 Not Found, if either the viewing session or MarkupBurner do not exist.
- 405 Method Not Allowed, if GET method is not used.

## Examples

### Example Request

```
GET http://localhost:18681/PCCIS/V1/ViewingSession/
uDLbVh9sTmXJAmD1GeXbS9Gn3WHxs8oib2xPsW2xEFjnIDdoJcudPtxciodSYFQq6zYGabQ_rJIecdbkImTTkSA/
MarkupBurner/5rGUUh3Qxhf6VXm8RkBPfA
```

```
200 OK
Content-Type: application/json
{
  "processId": "5rGUUh3Qxhf6VXm8RkBPfA",
  "state": "complete",
  "percentComplete": 100
}
```

## GET

/PCCIS/V1/ViewingSession/u{ViewingSessionID}/MarkupBurner/{processId}/Document?  
ContentDispositionFilename

Gets the output result of a MarkupBurner process for a viewing session.

### HTTP Method

GET

### Parameters

Name	Description	Details
ContentDispositionFilename	The filename, without extension, to use in the Content-Disposition header of the response (the file extension will automatically be added).  The default value is "document".	string, optional

### Request Body

None

### Status Codes

- 404 Not Found, which may occur if any of the following are true:
  - the MarkupBurner has not completed yet
  - no such MarkupBurner exists
  - no such viewing session exists
- 405 Method Not Allowed, if GET method is not used.

### Examples

#### Example Request

```
GET https://localhost:18681/PCCIS/V1/ViewingSession/u8091681f-15bf-4150-94a0-3ff7f2acd42d/MarkupBurner/ElkNzWtrUJp4rXI5YnLUgw/Document
```

#### Example Response

```
200 OK
```

## Multi-Server Mode

### GET /Service/Properties/Servers

Returns the list of PrizmDoc servers in a multi-server cluster that requests can be routed to. The list of servers returned here must first have been set by a request sent to PUT /PCCIS/V1/Service/Properties/Servers or via the Central Configuration File.

This URL is only available when PrizmDoc is running in Multi-Server Mode. See the [PrizmDoc Multi-Server Mode](#) topic for more details of this feature.

#### Http Method

GET

#### Resource URL

/PCCIS/V1/Service/Properties/Servers

#### Parameters

None

#### Request Body

None

#### Response Body

A JSON representation containing the IP address and port for all PrizmDoc servers in the current node.

#### Example

```
GET http://192.168.0.1:18681/PCCIS/V1/Service/Properties/Servers
```

#### Example Response

```
{
  "servers": [
    {
      "address": "192.168.0.1",
      "port": "18682"
    },
    {
      "address": "192.168.0.2",
      "port": "18682"
    },
    {
      "address": "192.168.0.3",
      "port": "18682"
    }
  ]
}
```

## PUT /Service/Properties/Servers

Sets the list of PrizmDoc servers in the multi-server cluster that requests should be routed to. This list of servers is used by each PrizmDoc server's Public Port to evenly spread out the requests for new viewing sessions across all available PrizmDoc servers, as well as route request for existing viewing session to the PrizmDoc server that originally created it.

The list of servers set with this request should also include the PrizmDoc server hosting the Public Port where this list is currently being set. For example, if this request is sent to the Public Port hosted at "192.168.0.1:18681", the list of servers should include the host and port of the Cluster Port on the same sever, "192.168.0.1:18682".

This URL is only available when PrizmDoc is running in multi-server mode. See the [PrizmDoc Multi-Server Mode](#) topic for more details of this feature.

### Http Method

PUT

### Resource URL

/PCCIS/V1/Service/Properties/Servers

### Parameters

None

### Request Body

A JSON representation containing the IP address and port for all PrizmDoc servers in the current node.

### Response Body

None

### 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"
    }
  ]
}
```

### Example Response

## Redaction Creator

### POST /RedactionCreator

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 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 /RedactionCreator/{id}](#) API 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.

#### Http Method

POST

#### Request Headers

Name	Value	Details
Accusoft-Affinity-Token	Affinity token returned in post response body for work file specified by input.documentFileId parameter in request body. Example: "rcqmuB9pAa8+4V7fhO1SXzawy/YMQU1g8lLdNDe5I7w="	Only required if PrizmDoc is running in <a href="#">multi-server mode</a> .

#### Parameters

None

#### Request Body

In the request body, provide JSON containing the following properties:

Name	Description	Details
input.documentFileId	The ID of the WorkFile that	string, required

	document whose text will be searched. This document will not be modified.	ek5Zb123oYHSUEVx1bUrVQ
input.autoRedactionRegularExpressions	<p>The Regular Expressions to match within the source document text.</p> <p>Multiple Regular Expressions provided in the array will be concatenated into a single Regular Expression using the format: "(regex1) (regex2)... (regexN)"</p>	<p>array, required</p> <p>Example: ["regex1", "regex2"]</p>
minSecondsAvailable	<p>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.</p>	<p>integer, optional</p> <p>Example: 60</p>

## Response Body

If successful, this method returns JSON containing the following properties:

Name	Description	Details
input.documentFileId	<p>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.</p>	<p>string</p> <p>Example: ek5Zb123oYHSUEVx1bUrVQ</p>
input.autoRedactionRegularExpressions	<p>The Regular Expressions to match within the source document text. This is an echo of what was passed in by the request.</p>	<p>array</p> <p>Example: ["[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"]</p>
expirationDateTime	<p>The date and time (in ISO 8601 Extended Format) when the RedactionCreator</p>	<p>string</p> <p>Example: "2014-05-13T20:38:39.796Z"</p>

processId	The ID of the RedactionCreator.	string Example: ElkNzWtrUJp4rXI5YnLUgw
state	The current state of the redaction creation process running on the server.  This will always be <b>"processing"</b> in this response.	string ("processing"   "complete"   "error" ) Example: "processing"
percentComplete	The percentage (0 – 100) complete of the redaction creation process.  This will always be <b>0</b> in this response.	integer Example: 0
errorCode	An error code string if a problem occurred during the redaction creation process.  This will always be <b>null</b> in this response.	string Example: null
output.markupFileId	The ID of the new WorkFile that represents the new redaction markup XML.  This will always be <b>null</b> in this response.	string
affiityToken	Affinity token echoed from request header. This value will only be present if PrizmDoc is running in <b>multi-server mode</b> .	string Example: "rcqmuB9pAa8+4V7fhO1SXzawy/YMQU1g8lLdNDe5I7w="

## Status Codes

- 200 OK, if RedactionCreator was created successfully.
- 400 Bad Request, if **input.documentFileId**, **input. autoRedactionRegularExpressions** is missing or invalid

- 405 Method Not Allowed, if POST HTTP method is not used.

## Examples

### Example Request

```
POST http://localhost:18681/PCCIS/V1/RedactionCreator
Content-Type: application/json
{
  "input": {
    "documentFileId": "ek5Zb123oYHSUEVx1bUrVQ",
    "autoRedactionRegularExpressions": [
      "[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"
    ]
  },
  "minSecondsAvailable": 60
}
```

### Example Response

```
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",
  "state": "processing",
  "percentComplete": 0,
  "errorCode": null,
  "output": null
}
```

## GET /RedactionCreator/{RedactionCreatorId}

Gets the state and result of an existing RedactionCreator.

Requests can be sent to thus 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.

### Http Method

GET

### Request Headers

Name	Value	Details
Accusoft-Affinity-Token	Affinity token returned in post response body for redaction creator specified by RedactionCreatorId parameter in URI. Example: "rcqmuB9pAa8+4V7fhO1SXzawy/YMQU1g8lLdNDe5I7w="	Only required if PrizmDoc is running in <b>multi-server mode</b> .

## Parameters

Name	Description	Details
RedactionCreatorId	The ID of the RedactionCreator.	string, required Example: ElkNzWtrUJp4rXI5YnLUgw

## Request Body

None

## Response Body

If successful, this method returns JSON containing the following properties:

Name	Description	Details
input.documentFileId	The ID of the WorkFile that represents the source document whose text will be searched. This document will not be modified.	string Example: ek5Zb123oYHSUEVx1bUrVQ
input.autoRedactionRegularExpressions	The Regular Expressions to match within the source document text.	array Example: ["[0-9]{3}[-]?[0-9]{2}[-]?[0-9]{4}"]
expirationDateTime	The date and time (in ISO 8601 Extended Format) when the MarkupBurner will be deleted.	string Example: "2014-05-13T20:38:39.796Z"
processId	The ID of the RedactionCreator.	string Example: ElkNzWtrUJp4rXI5YnLUgw
state	The current state of the redaction creation process running on the server.	string ("processing"   "complete"   "error" ) Example: "complete"

	<p>complete of the markup burning process.</p> <p><b>Note:</b> At time of writing, the percent complete will only ever be 0 or 100.</p>	Example: 100
errorCode	An error code string if a problem occurred during the redaction creation process.	string Example: "DocumentFileIdDoesNotExist"
output.markupFileId	The ID of the new WorkFile that represents a new Markup XML document specifying the redactions.	string Example: vry3FPE0zQqYwhzndRccOQ
affinityToken	Affinity token echoed from request header. This value will only be present if PrizmDoc is running in <b>multi-server mode</b> .	string Example: "rcqmuB9pAa8+4V7fhO1SXzawy/YMQU1g8lLdNDe5l7w="

## Status Codes

- 200 OK, if RedactionCreator was retrieved successfully.
- 404 Not Found, if **RedactionCreatorId** does not exist.
- 405 Method Not Allowed, if GET HTTP method is not used.

## Error Codes

- **DocumentFileIdError** - This indicates that an error occurred attempting to access the document via the WorkFile Service. The specified workfile Id is valid, but an error prevented the file from being read.
- **DocumentFileIdDoesNotExist** - This error indicates that the document requested by the specified workfile Id does not exist.
- **MarkupCreationError** - This error indicates that a markup file could not be created given the source document and regular expression(s).
- **MarkupWorkFileCreationError** - This error indicates that a workfile could not be created for the markup file.

## Examples

### Example Request

## Example Response

```
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",
  "state": "complete",
  "percentComplete": 100,
  "errorCode": null,
  "output": {
    "markupFileId": "vry3FPE0zQqYwhzndRccOQ"
  }
}
```

## Search Contexts

### Search Contexts

The *search context* and *search task* APIs are designed for a viewer 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".
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
Content-Type	Must be application/json
Accusoft-Affinity-Token	The affinityToken of the work file specified by <code>input.fileId</code> . <b>Required when server clustering is enabled and <code>input.source</code> is "workFile"</b> .

#### 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.

### 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.

- `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 errorCode	Description
400	"MissingInput"	Can occur when clustering is enabled and an Accusoft-Affinity-Token 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 /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/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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 /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

```
200 OK
Content-Type: application/json

{
  "input": {
    "documentIdentifier": "your-own-unique-identifier-for-the-source-document",
```

```
{
  "contextId": "ElkNzWtrUJp4rXI5YnLUgw",
  "state": "awaitingInput",
  "percentComplete": 0,
  "expirationDateTime": "2016-12-17T20:38:39.796Z",
  "affinityToken": "ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

## GET /v2/searchContexts/{contextId}

Gets information about a search context.

### 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. <b>Required when server clustering is enabled.</b>

### 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](#)

- `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.
500	"InternalError"	The server encountered an internal error when handling the request.

## Examples

### Example request

```
GET /v2/searchContexts/ElkNzWtrUJp4rXI5YnLUgw
Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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"
  },
}
```

```
    "affinityToken": "ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=",
  "state": "error",
  "errorCode": "InvalidPassword",
  "errorDetails": {
    "in": "searchContext",
    "at": "input.password"
  },
}
```

## 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/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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 contextId which identifies the resource.

#### Request Headers

Name	Description
Accusoft-Affinity-Token	The affinityToken of the search context. <b>Required when server clustering is enabled.</b>

### Successful Response

This request returns no body in the response when successful.

### Error Responses

Code	HTTP Error Code	Description
404	"Not Found"	No search context with the provided contextId could be found.
400	"MissingInput"	Can occur when clustering is enabled and an Accusoft-Affinity-Token request header was not provided.
500	"InternalServerError"	The server encountered an internal error when handling the request.

## Examples

### Example request

```
DELETE /v2/searchContexts/E1kNzWtrUJp4rXI5YnLUgw
Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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 contextId which identifies the resource.

### Request Headers

Name	Description
Content-Type	Must be application/json
Accusoft-Affinity-Token	The affinityToken of the search context. <b>Required when server clustering is enabled.</b>

### 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

- `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 errorCode	Description
400	"MissingInput"	Can occur when clustering is enabled and an Accusoft-Affinity-Token request header was not provided.
480	"MissingInput"	A required input value was not provided. See <code>errorDetails</code> in the

Code	HTTP error code	Description
		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 state of "error", or has otherwise become unusable.
480	"IncorrectUsage"	The state 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

### Example request

```
PUT /v2/searchContexts/ElkNzWtrUJp4rXI5YnLUgw/records
Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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 ],
      ...
    ]
  }, ...]
}
```

### Example 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": {
```

```
    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. <b>Required when server clustering is enabled.</b>

#### 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 <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

Code	HTTP error code	Description
		the response body.
580	"InternalError"	The server encountered an internal error when handling the request.

## Examples

### Example request

```
POST /v2/searchContexts/ElkNzWtrUJp4rXI5YnLUgw/completed
Accusoft-Affinity-Token: ejN9/kXEYOuken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

### Example 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",
  }
}
```

```
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}	<b>Required.</b> A set of comma-delimited page indices ( <b>zero-indexed</b> page numbers) and/or hyphenated page index ranges for which you want the full-page text data ( <i>records</i> ). See <a href="#">more below</a> .

#### *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

Accusoft-Affinity-Token

The affinityToken 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

- `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 errorCode	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 state 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 /v2/searchContexts/E1kNzWtrUJp4rXI5YnLUgw/records?pages=0-9
Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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": [
        [
```

```
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  
  
{
```

```
    "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 /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,
```

```
},
{
  "number": 2,
  "errorCode": "CouldNotGetPageData"
},
{
  "number": 3,
  "text": "and then, she said to the dragon...",
  "width": 612.00,
  "height": 792.00,
  "rectangles": [...]
}
]
```

## Search Tasks

### Search Tasks

The [search context](#) and [search task](#) APIs are designed for a viewer 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 <a href="#">search context</a> .
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 application/json
Accusoft-Affinity-Token	The affinityToken of the <a href="#">search context</a> specified by <code>input.contextId</code> . <b>Required when server clustering is enabled.</b>

#### Request Body

- `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.

## 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 errorCode	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.

Code		
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 /v2/searchTasks
Content-Type: application/json
Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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": "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](#).

## 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

## Request

### Request Headers

Name	Description
Content-Type	Must be application/json

### 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 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.

Code		
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
/v2/viewingSessions/DLbVh9sTmXJAmD1GeXbS9Gn3WHxs8oib2xPsw2xEFjnIDdoJcudPtxciodSYFQq6zYGabQ_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](#).

Gets information about a search task.

To get search results, use GET /v2/searchTasks/{processId}/results.

## Request

### URL Parameters

Parameter	Description
{processId}	The processId which identifies the search task.

### Request Headers

Name	Description
Accusoft-Affinity-Token	The affinityToken of the search task. <b>Required when server clustering is enabled.</b>

## 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 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 request header was not provided.
500	"InternalError"	The server encountered an internal error when handling the request.

## Example

### Request

```
GET /v2/searchTasks/pR5X6nPDgMwat6cx1mn0Q3
Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

### Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "input": {
    "contextId": "ElkNzWtrUJp4rXI5YnLUgw",
    "searchTerms": [{
      "type": "simple",
      "pattern": "quick",
      "caseSensitive": false,
      "contextPadding": 25
    }]
  },
}
```

```
"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 processId 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 affinityToken of the search task. <b>Required when server clustering is enabled.</b>

### 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

- `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 GET <code>/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 /v2/searchTasks/pR5X6nPDgMwat6cx1mn0Q3/results
Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "results": [
```

```
"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": "Cx07GHlkmI32gxAQhv49WZ",
"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 /v2/searchTasks/pr5X6nPDgMwat6cx1mn0Q3/results?continueToken=Cx07GHlkmI32gxAQhv49WZ
Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+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 }],
```

```
    "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 /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

{processId}	The processId which identifies the search task.
-------------	---

## Request Headers

Name	Description
Accusoft-Affinity-Token	The affinityToken of the search task. <b>Required when server clustering is enabled.</b>

## Successful Response

HTTP/1.1 204 No Content

## Errored 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.
500	"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": {
```

```
"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
    }]
  }
}
```

## System Information (Deprecated)

 The following API's have been deprecated and will be removed from the public API in a future release. They will not be replaced.

## GET Service/Current/Tasks

 This API has been deprecated and will be removed from the public API in a future release. It will not be replaced.

This API returns a JSON list of current tasks.

### Http Method

GET

### Resource URL

/PCCIS/V1/Service/Current/Tasks

### Parameters

None

### Request Body

Empty.

### Response Body

If successful, this method returns a list of objects with the following properties:

Property Name	Value	Description
id	String	The numeric ID of this task.
name	String	Information about the PrizmDoc license. <ul style="list-style-type: none"><li>• "not licensed"</li><li>• "licensed as '...'"</li></ul>
parameters	String	The parameters that were passed to the public API.
duration	Double	The time it took to run the task.

### Example

GET http://localhost:18681/PCCIS/V1/Service/Current/Tasks

### Example Response

```
[{"id":"1340","name":"GET Service Current Tasks", "parameters":"None","duration":0}]
```

## GET Service/History/Tasks

 This API has been deprecated and will be removed from the public API in a future release. It will not be replaced.

The information returned will span the last two minutes.

## Http Method

GET

## Resource URL

/PCCIS/V1/Service/History/Tasks

## Parameters

None

## Request Body

Empty.

## Response Body

If successful, this method returns a list of object with the following properties:

Property Name	Value	Description
id	String	The numeric ID of this task.
name	String	The name of the task. This will match, to some extent, the public API.
parameters	String	The parameters that were passed to the public API.
startTime	String	The UTC time when the task started, formatted as a string using ISO 8601. JSON does not have a formal date type so dates are represented as ISO 8601 strings.
duration	Double	The time it took to run the task.

## Example

GET http://localhost:18681/PCCIS/V1/Service/History/Tasks

## Example Response

```
[{"id":"2731","name":"GET Service History Tasks","parameters":"None", "startTime":"2013-08-14T14:23:49.5164460Z","duration":0.0080007999999999989}, {"id":"2732","name":"GET Service History Errors","parameters":"None", "startTime":"2013-08-14T14:23:49.8044748Z","duration":0}]
```

## GET Service/History/Errors

 This API has been deprecated and will be removed from the public API in a future release. It will not be replaced.

This API returns a JSON list of **HistoricalLogItem** for the last 15 minute time period.

GET

## Resource URL

/PCCIS/V1/Service/History/Errors

## Parameters

None

## Request Body

Empty.

## Response Body

If successful, this method returns a list of objects with the following properties:

Property Name	Value	Description
time	String	The UTC time when the event, formatted as a string using ISO 8601. JSON does not have a formal date type so dates are represented as ISO 8601 strings.
level	String	The NLog level of the item. This may be: <ul style="list-style-type: none"><li>• "Error"</li><li>• "Fatal"</li></ul>
message	String	The message associated with the event.
exception	String	Details of the exception that was thrown to cause this item to be logged. It is possible, but unlikely that this value will be NULL.

## Example

GET http://localhost:18681/PCCIS/V1/Service/History/Errors

## Example Response

```
[{"time":"2013-08-14T14:40:57.8242665Z","level":"Error","message": "End: Unable to process request","exception":"WebException: Unable to connect to the remote server"}, {"time":"2013-08-14T14:41:11.6906530Z","level":"Error","message": "Enterprise 3 document text watcher task failed","exception": "DirectoryNotFoundException: Could not find a part of the path \\u0027C:\\ProgramData\\Accusoft\\Prizm\\Cache\\ dfebd89e-6cde-43b3-a535-3a52accfe311\\25c79ba3-95dd-4907-94d1- ed3f91c8a3ec\\o11_rr150_Intermediate.xml\u0027."}]
```

## GET Service/History/ViewingSessions

 This API has been deprecated and will be removed from the public API in a future release. It will not be replaced.

## Http Method

GET

## Resource URL

/PCCIS/V1/Service/History/ViewingSessions

## Parameters

None

## Request Body

Empty.

## Response Body

If successful, this method returns a list of objects with the following properties:

Property Name	Value	Description
startTime	String	The UTC time when the event, formatted as a string using ISO 8601. JSON does not have a formal date type so dates are represented as ISO 8601 strings.
externalId	String	External reference ID string. Helpful for log tracing.
viewingSessionId	String	The document ID.

## Example

GET http://localhost:18681/PCCIS/V1/Service/History/ViewingSessions

## Example Response

```
[{"startTime":"2013-08-14T14:13:25.6740680Z","externalId": "F7-92-5C-CE-28-BE-AD-6F-79-C3-FE-FF-87-FE-FA-B6-73-7F-F5-92", "viewingSessionId":"7045a365-9f59-45eb-a1cf-7e213b5a6223"}, {"startTime":"2013-08-14T14:13:25.6740680Z","externalId": "F7-92-5C-CE-28-BE-AD-6F-79-C3-FE-FF-87-FE-FA-B6-73-7F-F5-92", "viewingSessionId":"0f0f0854-3019-4028-a576-27977dc01855"}, {"startTime":"2013-08-14T14:24:35.3830322Z","externalId": "F7-92-5C-CE-28-BE-AD-6F-79-C3-FE-FF-87-FE-FA-B6-73-7F-F5-92", "viewingSessionId":"bb2d91c7-4cf7-477b-a726-67349ee05a92"}]
```

## Viewing Sessions

To initiate a view, the following RESTful operations should be performed as detailed in the PrizmDoc Server [Sample](#) discussion.

### GET ViewingSession

provided in the POST request that created the viewing session.

## Http Method

GET

## Resource URL

/PCCIS/V1/ViewingSession/u{ViewingSessionID}

## Parameters

None

## Request Body

None

## Response Body

If successful, this method returns the same properties and values set in the body of the POST request that created the viewing session. Additionally, the following property is included in the response:

Property Name	Value	Description
creationTime	String	A UTC-formatted time string representing the moment the viewing session was created.

## Example

GET

```
http://localhost:18681/PCCIS/V1/ViewingSession/uGcIsIsEGbLV2_V9yy4NzmK2HB-JuLOH--A9sZ16cla9tx00ZDBGfq1G4kKu0r_GyEps4wWCvDwn4dpnZAR76Uw
```

## Example Response

HTTP 200

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,
}
```

```
sourceDocumentId": "0",  
"documentSource": null,  
"documentExtension": "help",  
"serverCaching": "full",  
"startConverting": null,  
"contentType": null,  
"pageContentEncryption": null,  
"watermarks": []  
}
```

 The "flash.optimizationLevel" property is kept for backward compatibility with PrizmDoc versions prior to v12.0.

## POST ViewingSession

The first step for viewing a document is to request a viewing session Identification JSON string. The body of the request is to send useful information about the document to the PrizmDoc Server which helps identify the document that will be uploaded to the service for conversion to a web compatible viewing format. The object to be posted here is the **ViewSessionProperties** resource which contains several properties.

The **origin** property is a key pair dictionary to contain any useful arbitrary data about the document. The PrizmDoc Server sample uses **ipAddress**, **hostname**, and **sourceDocument** for identification purposes, but any arbitrary data can be used that helps identify or characterizes the document.

The render property is important for storing information for client-side rendering. It stores rendering information for the HTML5 client. The `Html5RenderProperties` class object has properties to control aspects of SVG creation.

The **externalId** property and **tenantId** should also be set. Typically, the **externalId** is a unique value which makes it easier to connect items in the server log to that which is in the caller's log but is not used for any particular purpose and the same goes for **tenantId**. Please look at the PrizmDoc Server sample for more information on using this RESTful API.

### Http Method

POST

### Resource URL

/PCCIS/V1/ViewingSession

### Parameters

None

### Request Body

In the request body, supply a **ViewingSessionProperties** resource with the following optional properties:

Property Name	Value	Description
origin	List	A list of key-value pairs containing user-specific data. This is useful for tracking information pertinent to the request, like IP address,

render	Nested Object	The object that describes rendering options.
render.flash	Nested Object	Deprecated. Setting this object no longer has any effect.
render.flash.optimizationLevel	Integer	Deprecated. Setting this property no longer has any effect.
render.html5	Nested Object	The object that describes rendering options for the Viewer.
render.html5.alwaysUseRaster	Boolean	Forces PrizmDoc Server to always provide raster image data to the Viewer instead of SVG, even if the Viewer supports viewing SVG data.
render.html5.rasterResolution	Integer	Deprecated. Setting this property no longer has any effect.
render.html5.svgMaxImageSize	Number	<p>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.</p> <p>The default value for this property is configurable. The out-of-box configuration uses a default value of 8000.</p> <p>Use 0 to disable the optimization.</p>
render.html5.vectorTolerance	Number	<p>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.</p> <p>The default value for this property is configurable. The out-of-box configuration uses a default value of 0.3.</p> <p>Use 0 to disable the optimization.</p>
password	String	Specifies the password to open password-protected PDF documents.
serverCaching	String	<p>This optional property determines whether or not the conversion output is cached for potential reuse by other viewing sessions. Valid values are:</p> <ul style="list-style-type: none"> <li>• "full" means that the output will be cached on the server. (default)</li> <li>• "none" means that the output will not be cached on the server.</li> </ul> <p>By default, the output of the document conversion process is cached on the server for potential reuse in future viewing sessions</p>

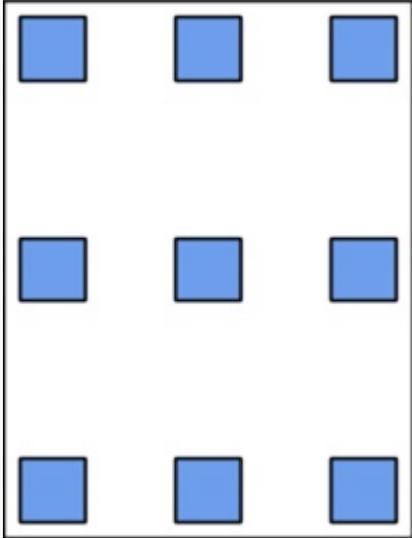
		is likely to be viewed multiple times across different viewing sessions. If this is not the case (e.g., a new viewing session is always a new document), you can save disk space on the server by explicitly setting the serverCaching property to "none" to prevent the caching of the output on the server.
watermarkText	String	Deprecated. Setting this property no longer has any effect.
externalId	String	<p>If the <b>documentSource</b> property is set to "api", empty, or is null, this property is not used by PrizmDoc Server and can be used to store a value of your choosing for the viewing session. This value could be useful for identifying requests in PrizmDoc Server log files or other purposes you find useful.</p> <p>If the <b>documentSource</b> property is set to "http" or "file", then this property must define the URL or local file path of the source document that PrizmDoc Server will retrieve. See the <a href="#">How to Transfer Your Document to PrizmDoc Server</a> topic for more details.</p>
tenantId	String	This is an ID for a tenant. Tenants are not a formal concept in this system and should be monitored and maintained in the calling system. However, having this data available will allow some optional behavior. For example, with this information, it would be possible to ensure some level of isolation between tenants.
attachmentIndex	Integer	Used for documents with attachment documents, like .MSG and .EML types. Otherwise, this value will typically be 0 for other document types.
attachmentDisplayName	String	Used for documents with attachment documents, like .MSG and .EML types. Provides the display name of the attachment.
countOfInitialPages	Integer	<p>The default value is 0, which will cause PrizmDoc Server to behave in its default manner converting all pages (running enterprise=3) of the document as soon as any page is requested.</p> <p>Setting <b>countOfInitialPages</b> to a value greater than 0 will instruct PrizmDoc Server to use two separate enterprise=3 operations: one to convert pages 0 through n-1 (where "n" the supplied value) and a second conversion to process pages "n" through the end of the document. The second conversion will only occur when content that has not been converted is requested. This should allow the first "n" pages to be created and delivered to the Viewer without forcing the entire document to be converted.</p>
documentSource	String	<p>This property tells PrizmDoc Server how it should expect to get the source document for a viewing session. The following values are supported:</p> <ul style="list-style-type: none"> <li>• "api" means that the document will be sent to PrizmDoc Server using the PUT HTTP request. This is the default and will be used if this property is null or empty.</li> <li>• "http" means that PrizmDoc Server should download the source document from the URL provided in the "externalId"</li> </ul>

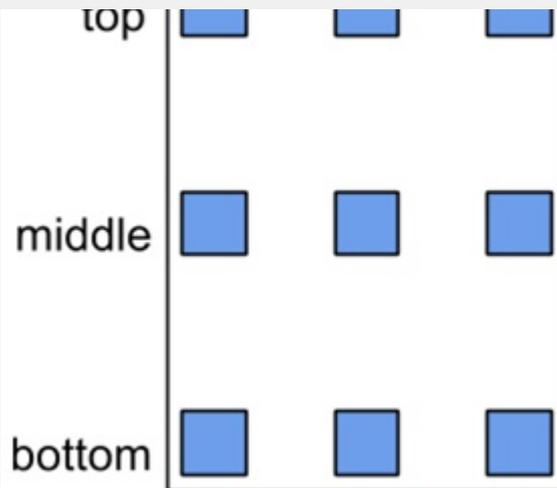
		<ul style="list-style-type: none"> <li>• "file" means that PrizmDoc Server should create a copy of the source document from the local file path provided in the "externalId" property.</li> </ul> <p>See the <a href="#">How to Transfer Your Document to PrizmDoc Server</a> topic for more details.</p>
documentExtension	String	<p>Defines the extension of the document specified in the <b>externalId</b> property. The Requirement of this property depends on whether <b>Format Detection</b> is enabled (default) or disabled.</p> <p>If Format Detection is disabled, then this property is only required if <b>documentSource</b> is "http" or "file" and the file name specified in <b>externalId</b> does not contain a valid document extension. If the file name specified in <b>externalId</b> contains a valid extension for the document, this property can be left unset.</p> <p>If Format Detection is enabled (default), then this property is optional with the following restrictions:</p> <ul style="list-style-type: none"> <li>• If the file is uniquely identified by the Format Detection process and no documentExtension is provided, the detected documentExtension will be used.</li> <li>• If the file is uniquely identified by the Format Detection process and a documentExtension is provided but differs from the detected documentExtension, the detected documentExtension will be used.</li> </ul> <p>If the file is not identified by the Format Detection process and a documentExtension is provided, the provided documentExtension will be used.</p> <p>The preceding "." should NOT be included in the extension value, otherwise an exception will be thrown and return 500 status from the POST HTTP request.</p>
startConverting	String	<p>This property determines when the process to generate pages for viewing is started. This property requires that <b>documentSource</b> is "http" or "file" and contentType is set to a valid value. Valid values are:</p> <ul style="list-style-type: none"> <li>• "initialPages" means the process to generate pages for viewing is started during the initial POST /ViewingSession request.</li> <li>• "none" means the process to generate pages for viewing is started later during a request to POST /ViewingSession/{id}/Notification/SessionStarted, GET /Page or GET /PageAttributes, whichever comes first.</li> </ul>
contentType	String	<p>Determines the kind of content we will start pre-generating in the background so that it is ready for request more quickly. In many cases, this property is optional (though it is required if documentSource is "http" or "file" and startConverting is</p>

		<p>Note that setting this property does not set the default content type for future requests for page content, nor does it limit in any way the types of content you may request. It is only a hint so that we can begin to prepare the kind of content you will likely soon request.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• "png"</li> <li>• "svg" - paths only, no inline font (not supported in IE8)</li> <li>• "svga" - paths optimized with an inline font (not supported in IE8, IE10)</li> <li>• "svgb" - paths further optimized with an inline font (not supported in IE8, IE9, IE10, desktop Safari 9.x or Chrome on Windows)</li> </ul>
serverSideSearch	String	<p>This property determines whether server-side text searching will be available for a document.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• "disabled" - server-side search will not be available for the viewing session.</li> <li>• "enabled" - server-side search will be available for the viewing session. (default)</li> </ul> <p>When enabled at viewing session creation, large document text retrieval and server-side searching via the search task API will be available. If disabled, requests to the search task API will result in an error.</p> <p>See the <a href="#">Search Tasks</a> topic for more details.</p>
minSecondsAvailable	Integer	<p>This optional property defines the amount of time in seconds for the viewingSession being created to be kept alive. The value is applicable only to viewing sessions created with the POST /ViewingSession request. A value of zero means the configured viewing session timeout value is used. The default viewingSession timeout is 20m (1200 seconds). This property is validated against the value set for viewing.sessionConstraints.minSecondsAvailable.max in the <a href="#">central configuration file</a>.</p> <p>When the serverCaching property is set to "full" and this property is provided, it must be 0 or a validation error will result. When the serverCaching property is set to "none" this property may be any positive integer up to viewing.sessionConstraints.minSecondsAvailable.max. If the value is less than the default viewing.sessionLifetime it will be set to the default viewing.sessionLifetime (20m).</p>
watermarks	Array	<p>The array of objects which describe the watermarks to apply to the viewing session. This object is optional. The presence of valid</p>

watermarks[n].type	String	<p>This property determines the type of watermark this object defines. The value of this property will determine what remaining properties are important according to the type.</p> <p>Valid values:</p> <ul style="list-style-type: none"><li>• "text" means that the current watermark object defines a watermark that will display horizontal text on all pages in the viewing session.</li><li>• "diagonalText" means that the current watermark object defines a watermark that will display diagonal text on all pages in the viewing session.</li><li>• "image" means that the current watermark object defines a watermark that will display an image on all pages in the viewing session.</li></ul> <p>Default value: N/A, this property is required.</p>
watermarks[n].opacity	Number	<p>This property determines how transparent or opaque a watermark appears over the page content. A value of 0.0 is completely transparent (not visible). A value of 1.0 is completely opaque.</p> <p>Valid values: 0.0 to 1.0</p> <p>Default value: 1.0</p>
watermarks[n].text	String	<p>This property determines the text string that will be displayed for "text" and "diagonalText" watermark types. This should not be set for "image" watermark types.</p> <p>Valid values: Any valid text string</p> <p>Default value: N/A, this property is required when type is "text" or "diagonalText".</p> <p>See the <a href="#">How to Watermark Content in a Viewing Session</a> topic for more information about special properties of the text string.</p>
watermarks[n].color	String	<p>This property determines the color of the text that will be displayed for "text" and "diagonalText" watermark types. This should not be set for "image" watermark types.</p> <p>Valid values: Any valid CSS color name (i.e. "red", "darkblue") or HEX value ("#FF0000")</p> <p>Default value: "black"</p>
watermarks[n].fontFamily	String	<p>This property determines the font name of the text that will be displayed for "text" and "diagonalText" watermark types. This should not be set for "image" watermark types.</p> <p>Valid values; Any valid font family name</p> <p>Default: Browser default font</p> <p>See the <a href="#">How to Watermark Content in a Viewing Session</a> topic for more information about the use of fonts.</p>

		<p>displayed for "text" and "diagonalText" watermark types. This should not be set for "image" watermark types.</p> <p>Valid values: A string containing a positive, non-zero decimal number followed by "pt" (i.e. "27.5pt").</p> <p>Default value: "12pt"</p>
watermarks[n].fontStyle	String	<p>This property determines the style of the text that will be displayed for "text" and "diagonalText" watermark types. This should not be set for "image" watermark types.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li>• "normal" means that the text will have no additional styling applied.</li> <li>• "italic" means that the text will be italicized.</li> </ul> <p>Default value: "normal"</p>
watermarks[n].fontWeight	String	<p>This property determines the weight of the text that will be displayed for "text" and "diagonalText" watermark types. This should not be set for "image" watermark types.</p> <p>Valid Values:</p> <ul style="list-style-type: none"> <li>• "normal" means that the text will be a normal weight.</li> <li>• "bold" means that the text will be bold weight.</li> </ul> <p>Default value: "normal"</p>
watermarks[n].textDecoration	String	<p>This property determines the decoration applied to the text that will be displayed for "text" and "diagonalText" watermark types. This should not be set for "image" watermark types.</p> <p>Valid Values:</p> <ul style="list-style-type: none"> <li>• "none" means that the text will have no additional decoration.</li> <li>• "underlined" means that the text will be underlined.</li> </ul> <p>Default value: "none"</p>
watermarks[n].horizontalAlign	String	<p>This property determines the horizontal alignment within the page for "text" and "image" watermark types. This property does not apply to "diagonalText" watermarks. For "left" and "right" values, sensible margins will be used so that the edge of the watermark does not fall on the very edge of the page.</p> <p>Valid Values:</p> <ul style="list-style-type: none"> <li>• "left" means that the watermark will be aligned on the left side of the page. Further, multi-line text watermarks will also have left alignment.</li> <li>• "center" means that the watermark will be aligned on the center of the page. Further, multi-line text watermarks will also have center alignment.</li> </ul>

		<p>side of the page. Further, multi-line text watermarks will also have right alignment.</p> <p>Default value: "center"</p> <p>left   center   right</p> 
watermarks[n].verticalAlign	String	<p>This property determines the vertical alignment within the page for "text" and "image" watermark types. This property does not apply to "diagonalText" watermarks. For "top" and "bottom" values, sensible margins will be used so that the edge of the watermark does not fall on the very edge of the page.</p> <p>Valid Values:</p> <ul style="list-style-type: none"><li>• "top" means that the watermark will be aligned on the top of the page.</li><li>• "middle" means that the watermark will be aligned on the middle of the page.</li><li>• "bottom" means that the watermark will be aligned on the bottom of the page.</li></ul> <p>Default value: "middle"</p>



watermarks[n].slope

String

This property determines the angle that the text is drawn for "diagonalText" watermark types. This should not be set for "image" and "text" watermark types.

Valid Values:

- "up" means that the diagonal text will start in the lower-left corner of the page and extend to the upper-right corner of the page.
- "down" means that the diagonal text will start in the upper-left corner of the page and extend to the lower-right corner of the page.

Default value: "up"

watermarks[n].autoSize

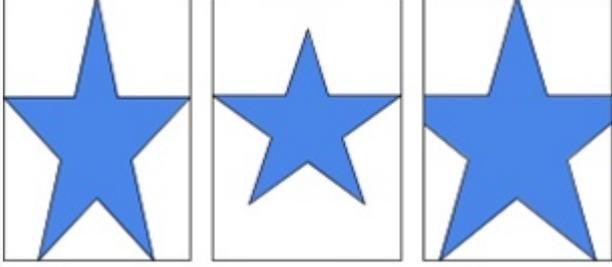
String

This property determines how an "image" watermark type will be sized to fill a page. If specified, any values for scale, verticalAlign, and horizontalAlign will be ignored. Note that the sensible automatic margins mentioned above for verticalAlign and horizontalAlign do not apply when auto-sizing an image watermark. In this case, the image should reach the very edge of the page. This should not be set for "text" and "diagonalText" watermark types.

Valid Values:

- "fit" means the image is scaled as far as possible while maintaining aspect ratio and without allowing any of the image to go beyond the edge of the page.
- "fill" means the image is scaled so that it completely fills the entire page with aspect ratio maintained. 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.
- "stretch" means the image is resized to be the same as the page. Aspect ratio is ignored.

Default value: N/A, the scale, verticalAlign, and horizontalAlign properties will be used if unset.

		
watermarks[n].scale	Number	<p>This property determines the size of images that are located using the verticalAlign and/or horizontalAlign properties. A scale value may be specified as a numeric value between 0.0 and 1.0. The value 1.0 indicates the image will be scaled to the size of the page and 0.0 indicates the image will be scaled infinitesimally small and will not be rendered. This should not be set for "text" and "diagonalText" watermark types.</p> <p>Valid values: 0.0 to 1.0</p> <p>Default value: 0.25 as long as autoSize is not set.</p>
watermarks[n].src	String	<p>This property determines the source of the image data for "image" watermark types. This should not be set for "text" and "diagonalText" watermark types.</p> <p>Note: The image watermark source must be a PNG. If other image formats are provided it will cause invalid watermarks to be created.</p> <p>Valid values:</p> <ul style="list-style-type: none"><li>• A work file ID referencing a valid work file. See the <a href="#">Work File</a> topic for more information about creating work files and obtaining their IDs.</li><li>• A URL specifying an absolute location with an HTTP or HTTPS scheme. This URL must be accessible from the server hosting the PrizmDoc RESTful Web Services.</li></ul> <p>Default value: N/A, required for "image" watermark types.</p>
pageContentEncryption	String	<p>Allows content encryption to be enabled or disabled for a single viewing session, overriding the PrizmDoc Server configuration on the server. See the "viewing.contentEncryption.enabled" setting in the <a href="#">Central Configuration Options</a> topic for more details about enabling or disabling content encryption on the server.</p> <p>Valid values for this property are:</p> <ul style="list-style-type: none"><li>• "enabled" means that the server will encrypt page content in the responses for the current viewing session. This value will override the PrizmDoc Server configuration setting.</li><li>• "disabled" means that the server will NOT encrypt page content in the responses for the current viewing session. This value will override the PrizmDoc Server configuration setting.</li><li>• "default" means that the PrizmDoc Server configuration setting will be used to determine whether or not it encrypts</li></ul>

session. This is the default behavior.

Setting this property to null or not including it at all means the same as "default".

See the topic [Enabling Content Encryption](#) for more information about this feature.

Additionally, the server configuration setting "viewing.sessionConstraints.pageContentEncryption.allowedValues" can be used to limit the values of this property that the server will accept. A value outside of the acceptable values defined by this server configuration setting will cause an error and the viewing session will not be created. See the "viewing.sessionConstraints.pageContentEncryption.allowedValues" setting in the [Central Configuration Options](#) topic for more details about defining allowable values for this property.

## Response Body

If successful, this method returns the following properties:

Property Name	Value	Description
viewingSessionId	String	The ID for this new viewing session.
affinityToken	String	The affinity token associated with this new viewing session, if PrizmDoc is running in Multi-Server Mode.

## Example

```
POST http://localhost:18681/PCCIS/V1/ViewingSession
```

## 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
    }
  },
  "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",
```

```
        "fontWeight": "bold",
        "verticalAlign": "bottom",
        "horizontalAlign": "right"
    }
]
}
```

## Example Successful Response

```
{
  "viewingSessionId": "8091681f-15bf-4150-94a0-3ff7f2acd42d"
  "affinityToken": " S2ZqtGi9vUAXBgdmM/PNNpCM4CApe9NxLIp/4QnAHlg="
}
```

## Examples of Errors Related to the minSecondsAvailable Property

Scenario when central configuration file contains: viewing.sessionConstraints.minSecondsAvailable.max: 1500

### Example

```
POST http://localhost:18681/PCCIS/V1/ViewingSession
Content-Type: application/json
{
  "tenantId": "my application name",
  "externalId": "my-unique-document-name.docx",
  "minSecondsAvailable": 3500
}
```

### Example Error Response

```
HTTP 480 InvalidInput
{
  "errorCode": "InvalidInput".
  "errorDetails": {
    "in": "body",
    "at": "minSecondsAvailable",
    "expected": {
      "type": "integer",
      "greaterThanOrEqualTo": 0,
      "lessThanOrEqualTo": 1500
    }
  }
}
```

## Example

```
POST http://localhost:18681/PCCIS/V1/ViewingSession
Content-Type: application/json
{
  "tenantId": "my application name",
  "externalId": "my-unique-document-name.docx",
  "minSecondsAvailable": 500,
  "serverCaching": "full"
}
```

## Example Error Response

```
HTTP 480 InputNotSupportedWithServerCachingEnabled
{
  "errorCode": "InputNotSupportedWithServerCachingEnabled".
  "errorDetails": {
    "in": "body",
    "at": "minSecondsAvailable",
  }
}
```

## DELETE ViewingSession/u{ViewingSessionID}

This RESTful API requests the PrizmDoc Server service to delete the viewing session.

### Http Method

DELETE

### Resource URL

/PCCIS/V1/ViewingSession/u{ViewingSessionID}

### Parameters

ViewingSessionID	The ID of the viewing session associated with the request.
------------------	--

### Request Body

None.

## Example

```
DELETE http://localhost:18681/PCCIS/V1/ViewingSession/u8091681f-15bf-4150-94a0-3ff7f2acd42d
```

### Response

When a DELETE request is made for a viewing session with serverCaching set to "none".

HTTP 204

When a DELETE request is made for a viewing session which does not exist.

### Example

HTTP 404

When a DELETE request is made for a viewing session with serverCaching set to "full".

### Example

```
HTTP 580
Content-Type: application/json
{
  "errorCode": "CannotDeleteCachedViewingSession"
}
```

## PUT ViewingSession/u{ViewingSessionID}/SourceFile?FileExtension={FileExtension}

A document source is uploaded to the PrizmDoc Server by writing the contents into the request stream using the viewing session ID from the previous viewing session ID request.

### Http Method

PUT

### Resource URL

/PCCIS/V1/ViewingSession/u{ViewingSessionID}/SourceFile?FileExtension={FileExtension}

### Parameters

ViewingSessionID	The ID of the viewing session associated with the request.
FileExtension	<p>This is the 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, 'jpg' is accepted but '.jpg' will return a 400 HTTP status). Extensions are not case sensitive. If Format Detection is disabled, the FileExtension must be provided.</p> <p>If Format Detection is enabled (default), the use of the FileExtension is as follows:</p> <ul style="list-style-type: none"><li>• If the file is uniquely identified by the Format Detection process and no FileExtension is provided, the detected FileExtension will be used.</li><li>• If the file is uniquely identified by the Format Detection process and a FileExtension is provided but differs from the detected FileExtension, the detected FileExtension will be used.</li><li>• If the file is not identified by the Format Detection process and a FileExtension is provided, the provided FileExtension will be used.</li></ul>

is provided, an error code of `UnrecognizedFormat` will be returned with a HTTP 580 status.

## Request Body

In the request body, supply the document data to be uploaded.

## Response Body

Empty.

### Example

```
PUT http://localhost:18681/PCCIS/V1/ViewingSession/u8091681f-15bf-4150-94a0-3ff7f2acd42d/SourceFile?FileExtension=doc
```

### Example Response

Empty

## GET ViewingSession/u{ViewingSessionID}/SourceFile{?ContentDispositionFilename}

Returns the original source document for a valid, active viewing session. The document returned will be an identical copy of the document originally provided to PCCIS; no conversions to other formats are supported.

The response will set the Content-Type header value to the registered MIME type associated with document extension of the original document. If a registered MIME type is not found, the value "application/octet-stream" is used.

## Http Method

GET

## Resource URL

/PCCIS/V1/ViewingSession/u{ViewingSessionID}/SourceFile

## Parameters

Name	Description	Details
ViewingSessionID	ID of the viewing session.	string, required Example: uW1ZpNTSfU139Zh3dfdhwbf
ContentDispositionFilename	Name to use for the filename attribute in the Content-Disposition response header. The default value is "file-{WorkFileId}. {extension}". The file extension of the source file will automatically be added to the Content-	string, optional Example: MonthlySalesReport

## Request Body

None

## Response Body

The document data, in its original format.

### Example

```
GET http://localhost:18681/PCCIS/V1/ViewingSession/u8091681f-15bf-4150-94a0-3ff7f2acd42d/SourceFile
```

### Example Response

```
200 OK
Content-Type: application/msword
Content-Disposition: attachment; filename=MontlySalesReport.pdf
[Document Data]
```

## POST ViewingSession/u{ViewingSessionID}/Notification/SessionStarted

This RESTful API requests the PrizmDoc Server service to begin conversion of the document to HTML format. The object property, viewer, will be set to "HTML5".

 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.

## Http Method

POST

## Resource URL

/PCCIS/V1/ViewingSession/u{ViewingSessionID}/Notification/SessionStarted

## Parameters

ViewingSessionID	The ID of the viewing session associated with the request.
------------------	--

## Request Body

In the request body, supply an object with the following optional property:

Property Name	Value	Description
viewer	String	Specify the type of viewer being used. The only supported value is "HTML5". The default value is "HTML5".

## Response Body

Empty.

### Example

GET /PCCIS/V1/ViewingSession/u{ViewingSessionID}/Notification/SessionStopped

## Example Response

Empty

## POST ViewingSession/u{ViewingSessionID}/Notification/SessionStopped

This request is used to stop, or invalidate, an active viewing session. This can be useful in situations when a viewing session is started but something prevents the document from being uploaded to the PrizmDoc RESTful Web Services successfully. You may also have a need in your application to stop a viewing session after a user is done with it and before the viewing session expires.

The parameters provided in the body of the request should be set with some consideration within your application. The `httpStatus` and `endUserMessage` property values specified in the body of this request will determine the HTTP status code and reason phrase that will be returned in the response for any request of the viewing session once this request has been issued.

For example, if this request is sent with `httpStatus = 504` and `endUserMessage = "Session is no longer available"` and then a request is made to get page 0 for the same viewing session, the response status of that request will be 504 Session is no longer available and the page content will not be delivered.

This allows you to explicitly handle the error case when requests are sent after the session has been stopped.

### Http Method

POST

### Resource URL

/PCCIS/V1/ViewingSession/u{ViewingSessionID}/Notification/SessionStopped

### Parameters

ViewingSessionID	The ID of the viewing session associated with the request.
------------------	--

### Request Body

In the request body, optionally supply a **SessionStoppedProperties** resource with the following optional properties:

Property Name	Value	Description
endUserMessage	String	A message suitable for display to the end user to be returned if any attempt to use the session is made after the session has been stopped. Default is "Session is stopped".
httpStatus	Integer	The http status suitable for display to the end user to be returned if any attempt to use the session is made after the session has been stopped. Default is 580.
accusoftErrorNumber	Integer	A custom Accusoft error number that should be sent back to the end user to be returned if any attempt to use the session is made after the

serverLogMessage	String	A message that should be placed into the server's log file when the session is stopped. Default is "The viewing session is stopped on request from the client".
------------------	--------	---

## Response Body

Empty.

### Example

```
POST http://localhost:18681/PCCIS/V1/ViewingSession/u8091681f-15bf-4150-94a0-3ff7f2acd42d/Notification/SessionStopped
{
  "endUserMessage": "Session no longer available",
  "httpStatus": 504,
}
```

### Example Response

200 OK

## PUT /PCCIS/V1/ViewingSession/u{id}/SourceRef

Assigns an existing **work file** to be used as the source document for this viewing session. This can be particularly useful when you want to avoid repeatedly uploading the same source file to the back end.

### Request

#### Query Parameters

None

#### Request Headers

None

#### Request Body

A JSON object specifying a refType of "workFile" and the file id of the work file to use.

Name	Description	Details
refType	Must be set to "workFile".	string, required
fileId	The id of the work file to use as a source document.	string, required

### Example

```
{
  "refType": "workFile",
  "fileId": "mUiXiqsQuevJKO9Swa32Bd"
}
```

### Successful Response

200

## Response Headers

*None*

## Response Body

*Empty*

## Error Response: Work file not found

If the work file you specified has expired or does not exist, you will receive an HTTP 480 response with a JSON object containing an errorCode of "NotFound".

## HTTP Status Code

480

## Response Headers

Name	Description
Content-Type	Will be set to "application/json".

## Response Body

### Example

```
HTTP 480
Content-Type: application/json
{
  "errorCode": "NotFound",
  "errorDetails": {
    "in": "body",
    "at": "fileId"
  }
}
```

## Error Response: Cannot change document once set

Once a viewing session is given a source document, that document cannot be changed. If a viewing session already has a source document and you attempt to assign a document by reference, you will receive an HTTP 480 error response with an errorCode of "CannotChangeDocument":

### Example

```
HTTP 480
Content-Type: application/json
{
  "errorCode": "CannotChangeDocument",
  "errorDetails": {
    "in": "body",
    "at": "fileId"
  }
}
```

## GET /PCCIS/1/ViewingSession/{id}/FileId

Returns the id of the work file which is being used for this viewing session.

Regardless of how a source document is provided, whether it is uploaded directly or whether an existing work file is used by reference, once the viewing engine has acquired the source document it will first ensure that a copy of it is present in the **work file** system.

Getting the work file id for a viewing session can be particularly helpful if you have uploaded a file directly to a viewing session and then want to reuse that file for a new viewing session without uploading it again.

### Request

#### Query Parameters

None

#### Request Headers

None

#### Response

#### HTTP Status Codes

200

480 - When a document has not been provided yet.

#### Response Headers

Name	Description
Content-Type	Will be set to "application/json".

#### Response Body

When a source document exists:

As long as this viewing session has a source document, the response will contain a work file id:

```
Example
HTTP 200
Content-Type: application/json
{
  "fileId": "mUiXiqsQuevJKO9Swa32Bd"
}
```

When a source document has not been provided:

If you just created a viewing session and have not yet provided a source document (either by upload or reference), then this URL will respond with:

```
Example
HTTP 480
Content-Type: application/json
{
```

```
    "in": "body",  
    "at": "fileId"  
  }  
}
```

## Work Files

The work file service is a temporary storage system which allows you 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

<code>POST /PCCIS/V1/WorkFile</code> when body is file bytes	Creates a work file resource from file bytes (used to <b>upload</b> input files).
<code>POST /PCCIS/V1/WorkFile</code> 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).
<code>GET /PCCIS/V1/WorkFile/{fileId}</code>	Gets the file bytes associated with a work file resource (used to <b>download</b> output files).
<code>GET /PCCIS/V1/WorkFile/{fileId}/Info</code>	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 /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/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="  
}
```

## 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 /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> ) or when <code>automatic format detection</code> has been explicitly disabled. 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.	pdf
MinSecondsAvailable	integer	The minimum number of seconds this work file must remain available. The actual lifetime may be longer.	300

## 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/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

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 <b>Accusoft-Affinity-Token</b> request header when submitting the POST to create the other resource.

## Error Responses

Status Code	JSON errorCode	Description
400	-	There was a problem with an input parameter: <ul style="list-style-type: none"><li>• FileExtension was required and not provided</li><li>• FileExtension was an invalid value</li><li>• MinSecondsAvailable was not a number</li></ul>
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 FileExtension. If you are getting this error for every file type which you upload, it is possible that <b>automatic format detection</b> needs to be enabled on your server.

## 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 /PCCIS/V1/WorkFile
Content-Type: application/json
Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=

{
  "primaryPath": "master.dwg",
  "items": [
    { "fileId": "CVBuD7DbQYNoJDqByGierQ", "path": "master.dwg" },
    { "fileId": "5qTYa3gzN9gYUb5SszqUhgq", "path": "parts/a.dwg" },
    { "fileId": "o1bLJwFGxf9QGuTkyr0qig", "path": "parts/b.dwg" }
  ]
}

HTTP/1.1 200 OK
```

```
{
  "fileId": "nkG9fiAmj27X3MhqGdbsXA",
  "primaryPath": "master.dwg",
  "items": [
    { "fileId": "CVBuD7DbQYNoJDqByGierQ", "path": "master.dwg" },
    { "fileId": "5qTYa3gzN9gYUb5SzqUhgq", "path": "parts/a.dwg" },
    { "fileId": "o1bLJwFGxf9QGutkyr0qig", "path": "parts/b.dwg" }
  ],
  "affinityToken": "ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

## Request

### Request Headers

Name	Description
Content-Type	You <b>must</b> 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 Content-Type 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 /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": "5qTYa3gzN9gYUb5SzqUhgq", "path": "parts/a.dwg" },
    { "fileId": "o1bLJwFGxf9QGutkyr0qig", "path": "parts/b.dwg" }
  ]
}
```

Name	Type	Description
primaryPath	string	<b>Required.</b> The primary entry point for viewing or conversion. This value must match the path value of one of the objects in the items array.
items	array	<b>Required.</b> The work files which are to be included in the set and what their local paths would be.

## items

Name	Type	Description
fileId	string	<b>Required.</b> The fileId of a work file to be included in the set.
path	string	<b>Required.</b> 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 path 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
Content-Type	application/json

## Response Body

```
{
  "fileId": "nkG9fiAmj27X3MhqGdbsXA",
  "primaryPath": "master.dwg",
  "items": [
    { "fileId": "CVBuD7DbQYN0JDqByGierQ", "path": "master.dwg" },
    { "fileId": "5qTYa3gzN9gYUb5SzqUhgq", "path": "parts/a.dwg" },
    { "fileId": "o1bLJwFGxf9QGuTkyr0qig", "path": "parts/b.dwg" }
  ],
  "affinityToken": "ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

Name	Type	Description
fileId	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.
primaryPath	string	The primary entry point which will be used for viewing or conversion.
items	array	The work files which are included in the set and their assigned paths.
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

480	"MissingInput"	A required input was not provided. See the errorDetails in the response.
480	"InvalidInput"	One of the input values was invalid. Possible causes: <ul style="list-style-type: none"><li>• MinSecondsAvailable was not a number.</li><li>• primaryPath did not match the path of any of the items.</li></ul> See the errorDetails in the response.
480	"ValueMustBeUnique"	One or more of the path values in the items array was non-unique.
580	UnrecognizedFileFormat	This will occur if you forget to specify a Content-Type of application/json when making your request. If no Content-Type is specified (or if any value other than application/json 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 Content-Type header to application/json.

## GET /PCCIS/V1/WorkFile/{fileId}

Gets the file bytes associated with a work file resource (used to download output files).

Example:

```
GET /PCCIS/V1/WorkFile/Xe6zv3dH0kVSzLuaNhd32A
Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

```
HTTP/1.1 200 OK
Content-Type: application/pdf
```

<<file bytes>>

### Request

#### Request Headers

Name	Description
Accusoft-Affinity-Token	If there is more than one machine in the cluster (an affinityToken was provided in the response to the initial POST request to create the work file), then this header must be set to the affinityToken of the work file you are trying to access.

#### Request Parameters

```
GET /PCCIS/V1/WorkFile/{fileId}{?ContentDispositionFilename}
```

fileId	string	<b>Required.</b> The fileId of the work file whose file bytes you want to download.	Xe6zv3dH0kVSzLuaNhd32A
ContentDispositionFilename	string	<p>The filename which will be used in the response Content-Disposition header, but without any file extension. The file extension will be supplied for you automatically.</p> <p>For example, if you specified a ContentDispositionFilename of FinancialReport and the fileExtension for the work file was "pdf", then the response would contain the following header:</p> <p>Content-Disposition: attachment; filename=FinancialReport.pdf</p> <p>If you do not specify a value, a filename will be used which is based upon the fileId of the work file, for example:</p> <p>Content-Disposition: attachment; filename=file-Xe6zv3dH0kVSzLuaNhd32A.pdf</p>	FinancialReport

## Successful Response

The raw bytes of the file.

## Response Headers

Name	Description	Example
Content-Type	The MIME type of the file being downloaded.	application/pdf
Content-Disposition	<p>Set to attachment with a filename based on the ContentDispositionFilename parameter or the fileId and using the fileExtension for this work file.</p> <p>See the ContentDispositionFilename request parameter.</p>	attachment; filename=file-Xe6zv3dH0kVSzLuaNhd32A.pdf

## Response Body

The raw bytes of the file.

## Error Responses

Status Code	JSON errorCode	Description
400	"InvalidInput"	<p>One of the input values was invalid.</p> <p>Possible causes:</p> <ul style="list-style-type: none"> <li>Accusoft-Affinity-Token was invalid.</li> </ul> <p>See the errorDetails in the response.</p>

480	"DataNotAvailable"	No file bytes exist for download. This can occur if you attempt to download the file bytes for a "package" work file which merely groups a set of related work files under a single fileId.
404	-	No work file existed for the given fileId.
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 /PCCIS/V1/WorkFile/Xe6zv3dH0kVSzLuaNhd32A/Info
Accusoft-Affinity-Token: ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM=
```

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "fileId": "Xe6zv3dH0kVSzLuaNhd32A",
  "fileExtension": "pdf",
  "affinityToken": "ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
}
```

## Request

### Request Headers

Name	Description
Accusoft-Affinity-Token	If there is more than one machine in the cluster (an affinityToken was provided in the response to the initial POST request to create the work file), then this header must be set to the affinityToken of the work file you are trying to access.

### Request Parameters

```
GET /PCCIS/V1/WorkFile/{fileId}/Info
```

Name	Type	Description	Example
fileId	string	<b>Required.</b> The fileId 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": "XEOZVJmIKRVJZLUdmUJZA",  
"fileExtension": "pdf",  
"affinityToken": "ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
```

Name	Type	Description
fileId	string	The unique id of this resource.
fileExtension	string	File extension assigned to this resource, indicating 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.

#### For "package" work files:

```
{  
  "fileId": "nkG9fiAmj27X3MhqGdbsXA",  
  "primaryPath": "master.dwg",  
  "items": [  
    { "fileId": "CVBuD7DbQYNoJDqByGierQ", "path": "master.dwg" },  
    { "fileId": "5qTYa3gzN9gYUb5SzqUhgq", "path": "parts/a.dwg" },  
    { "fileId": "o1bLJwFGxf9QGuTkyr0qig", "path": "parts/b.dwg" }  
  ],  
  "affinityToken": "ejN9/kXEY0uken4Pb9ic9hqJK45XIad9LQNgCgQ+BkM="
```

Name	Type	Description
fileId	string	The unique id of this resource.
primaryPath	string	The primary entry point which will be used for viewing or conversion.
items	array	The work files which are included in the set and their assigned paths.
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.

#### Error Responses

Status Code	Description
404	No work file existed for the given fileId.
405	An HTTP method other than GET was used.

## PrizmDoc E-Signature Viewer API

## Developer Reference

- PrizmDoc E-Signature Viewer API
  - [Developer Reference](#)
  - [Class: ESigner](#)
  - [Class: TemplateDesigner](#)
  - [External: jQuery.fn](#)
  - [Module: button-set](#)
  - [Module: checkbox-collection](#)
  - [Module: data-persist](#)
  - [Module: date-picker](#)
  - [Module: download-signed-form](#)
  - [Module: download-signed-form-trigger](#)
  - [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](#)

## Class: ESigner

## 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: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`

*Type:*

- `module:event-store`

`fillProgress` :`module:fill-progress`

*Type:*

- `module:fill-progress`

`formController` :`module:form-controller`

*Type:*

- `module:form-controller`

`formExtraction` :`module:form-extraction`

*Type:*

- `module:form-extraction`

`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 3.3.3](#) on Mon Mar 13 2017 16:48:42 GMT-0400 (Eastern Daylight Time)*

## Class: TemplateDesigner

Class: TemplateDesigner

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: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:

`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`

## destroy()

Destroys the viewer and all modules, and returns the parent DOM element to its original state.

---

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:43 GMT-0400 (Eastern Daylight Time)

## 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			<p>The viewingSessionId to use in ViewerControl.</p> <p>This identifier is generated by PCCIS. (For version 1 of the PCCIS REST API, see the service POST /PCCIS/V1/ViewingSession.)</p> <p>Sample web tiers that ship with the Viewer demonstrate how to generate a viewing session and pass it to the viewer.</p> <p><b>This value is always required.</b></p>
templateDocumentId	string		<optional>	<p>This parameter will be used when creating a template in order to associate the template form with a document. When opening a form, the value of this parameter will be used to identify the document associated with the</p>

			formDefinition. <b>This value is required by the TemplateDesigner.</b>
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 for edition, while leaving the value out will cause it to create a new form. <b>This value is required by the ESigner.</b>
formRoleId	string	<optional>	The formRoleId to be used when opening a form. Specifying this value in the ESigner will cause it to only create fields assigned to the given form role.
signatureDateFormat	external:"jQuery.fn"~DateFormat	<optional>	"MM/DD/YYYY" Specifies the date format that is stored when saving a template. When the e-signer loads a template, it displays date signatures using the date format stored in the template.
language	object		Specifies the language to use for the text in the viewer. Use this option to localize the viewer.  This property should be set to the contents of the file "viewer-assets/languages/en-US.json". Both the e-signer and the template-designer ship with their own version of the language file. Each viewer has a different set of language strings in the language file. <b>This value is always required.</b>
imageHandlerUrl	string	<optional>	"../pcc.ashx" The end point of the web tier services that support the viewer.
onViewerCreation	function	<optional>	This function is available for both the ESigner and the TemplateDesigner, it will trigger when the viewer's DOM is ready to use. You can use one parameter inside this function, it will be an ESigner or a TemplateDesigner object

depending on which Viewer 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.

	<b>Token</b>	<b>Output</b>
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:**

<b>Name</b>	<b>Type</b>	<b>Description</b>
options	<a href="#">external:"jQuery.fn"~configParameters</a>	The configuration options used to initialize the viewer.

See: [ESigner](#) for an example.

A jQuery plugin to initialize a new [TemplateDesigner](#) viewer.

**Parameters:**

Name	Type	Description
options	<a href="#">external:"jQuery.fn"~configParameters</a>	The configuration options used to initialize the viewer.

See: [TemplateDesigner](#) for an example.

---

Documentation generated by [JSDoc 3.3.3](#) on Mon Mar 13 2017 16:48:42 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.

`(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:**

Type

HTMLElement

### 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>
```

```
// 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);
});
```

## 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**

**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
<code>val</code>	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 3.3.3* on Mon Mar 13 2017 16:48:42 GMT-0400 (Eastern Daylight Time)

## Module: checkbox-collection

### Module: checkbox-collection

A set of UI checkboxes that interact as one group.

**(require("checkbox-collection"))(e1) → {HTMLElement}**

Parses and initializes a checkbox collection.

Name	Type	Description
e1	HTMLElement	The parent element in which to parse for the checkbox collection component.

### Returns:

The parsed checkbox collection element.

Type

HTMLElement

### 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
{
  CheckboxCollection(this);
});
```

**(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

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 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.**

#### *Listens to Events:*

- `module:event-store#event:PersistSignatures`

#### *Example*

```
var DataPersist = require('data-persist.js');

// a generic Viewer constructor
var myDataPersist =
  DataPersist(viewer);
```

```
(require("data-persist"))(viewer)
```

Created the data persistence module.

#### *Parameters:*

Name	Type	Description
viewer	Core	The core viewer to which the module will attach.

#### Methods

`destroy()`

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 GMT-0400 (Eastern Daylight Time)

## Module: date-picker

### Module: date-picker

Provides a date picker UI.

`(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 abject.

Properties		
Name	Type	Description
dateFormat	external:"jQuery.fn"~DateFormat	The format string to use when providing the selected date.

#### 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
```

```
{
  dateFormat: 'MM/DD/YYYY'
});
```

## Methods

### destroy()

Destroys the module.

---

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 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.

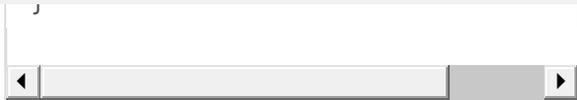
#### **Listens to Events:**

- `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('myDownloadSig
```



```
(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.

**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 3.3.3](#) on Mon Mar 13 2017 16:48:42 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.

```
(require("download-signed-form-trigger"))(viewer,
```

Creates the download signed form trigger UI module.

**Parameters:**

Name	Type	Description
viewer	Core	The core viewer to which the module will attach.
options	Object	An options object.

**Properties**

Name	Type	Description
elem	HTMLElement	The element in which the module UI will be inserted.

**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('myDownloadSig

    });
}
```

**Methods**

`destroy()`

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 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
```

```
<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`

**Type:**

- `module:event-store~on`

**Methods**

`addOption(val, styleopt) → {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.

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 3.3.3* on Mon Mar 13 2017 16:48:42 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);
}
```

### Members

(inner) **onDoneCallback**

The format of the data provided in the onDone event. This will be an event triggered in the style of `module:event-`

### Properties:

Name	Type	Description
status	string	On a successful completion, this value will be success. It can also be cancel for when a user cancels the action.
data	*	The data requested by the action. This can be any format that is needed by the event.

### Methods

`(inner) eventCallback(event, dataopt)`

All event callbacks across the viewer follow this pattern.

### Parameters:

Name	Type	Attributes	Description						
event	Object		The event parameters.						
<b>Properties</b>									
<table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>type</td><td>string</td><td>The event name.</td></tr></tbody></table>				Name	Type	Description	type	string	The event name.
Name	Type	Description							
type	string	The event name.							
data	Object	<optional>	The data associated with this event. This data object will contain the information being forwarded with the event.						

`(inner) off(nameopt, callbackopt)`

All event removals across the viewer follow the pattern of this off function.

### Parameters:

Name	Type	Attributes	Description
name	string	<optional>	The name of the event being listened to. If a name is not provided, all event listeners attached to this object will be removed.
callback	eventCallback	<optional>	The original function

used to register the event listener that should be disconnected. If this parameter is not defined, all functions registered to listen to the provided event name will be removed.

(inner) `on(name, callback)`

All event registrations across the viewer follow the pattern of this on function.

**Parameters:**

Name	Type	Description
name	string	The event name.
callback	eventCallback	The function to execute.

## 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](#)

```
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.  <b>Properties</b> <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.																
onDone	string	<optional>	An event name to trigger, in the style of <a href="#">module:event-store~onDoneCallback</a> . It should provide a date string as the event's data attribute.															

[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 <a href="#">module:event-store~onDoneCallback</a> . It should provide a signature object as the data 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('DeleteField',
{
  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	<a href="#">module:state-store~FormDefinition</a>	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.
});
```

### DuplicateFields

Triggers duplication of the given fields.

### Properties:

markIds    Array    An array of Mark IDs to duplicate.

See:                    [module:event-store~eventCallback](#)

### Example

```
viewer.eventStore.trigger('DuplicateFi
{
  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 <a href="#">module:event-store~onDoneCallback</a> . It should provide a date string as the event's data attribute.

See:                    [module:state-store~FieldList](#)

[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.  <b>Properties</b> <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](#)

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 data 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('MatchSizeFi
{
  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.
operation	string	<optional>	"extend"	Specifies how the modification should occur. By default, any modification will

extend the current state, merging any additional values from stateValue 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 stateValue 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:

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:

state	string	This should always be defined as the string "KeyCombinations".
stateValue	Object	An object containing data about the key combination.
<b>Properties</b>		
Name	Type	Description
keyCombinations	string	Keyboard key combinations when pressed would trigger <a href="#">KeyCombinationsTriggered</a> 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](#)

## 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('TemplateSav
```

### TemplateSaveFailed

Indicates that a template failed to save.

See: [module:event-store~eventCallback](#)

### Example

```
viewer.eventStore.trigger('TemplateSav
```

### ToggleChecklist

Triggers the checklist to toggle open or closed.

See: [module:event-store~eventCallback](#)

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 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.

`(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		
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>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

```
edit.js');  
  
// a generic Viewer constructor  
function Viewer(opts) {  
    var myFieldEdit =  
    FieldEdit(this, {  
        elem:  
        document.getElementById('myFieldEdit')  
  
    });  
}
```

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 GMT-0400 (Eastern Daylight Time)

## Module: field-list

### Module: field-list

Manages a list of fields and allows drag and drop reordering.

`(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**

Name	Type	Description
elem	HTMLElement	The element in which the module UI will be inserted.

#### Fires:

- `module:event-store#event:ModifyTemplateField`

- `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')  
  });  
}
```

### Methods

#### `destroy()`

Destroys the module.

---

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 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`

```
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**

Name	Type	Description
elem	HTMLElement	The element in which the module UI will be inserted.

#### Methods

**destroy()**

Destroys the module.

---

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 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.

#### ***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:FormLoaded`
- `module:event-store#event:BurnForm`

#### ***Example***

```
var FillFormController =
  require('fill-form-controller.js');

// a generic Viewer constructor
function Viewer(opts) {
  var myFillFormController =
    FillFormController(this);
}
```

```
(require("fill-form-controller"))(viewer)
```

Creates the form controller module.

#### ***Parameters:***

viewer	Core	The core viewer to which the module will attach.
--------	------	--

## Methods

**destroy()**

Destroys the module.

---

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 GMT-0400 (Eastern Daylight Time)

## Module: fill-main-toolbar

Module: fill-main-toolbar

Manages the form's main toolbar.

`(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**

Name	Type	Description
elem	HTMLElement	The element in which the module UI will be inserted.

### 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('myFillMainToo  
  
    });  
}
```

### Methods

**destroy()**

Destroys the module.

---

Documentation generated by [JSDoc 3.3.3](#) on Mon Mar 13 2017 16:48:42 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.

**(require("fill-progress"))(viewer, options)**

Creates the fill progress UI module.

#### **Parameters:**

viewer	Core	The core viewer to which the module will attach.
options	Object	An options object.
<b>Properties</b>		
<b>Name</b>	<b>Type</b>	<b>Description</b>
elem	HTMLElement	The element in which the module UI will be inserted.

**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');
  });
}
```

**Methods**

`destroy()`

Destroys the module.

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 GMT-0400 (Eastern Daylight Time)

## 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.

```
(require("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: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:ModifyTemplateField`
- `module:event-store#event:FormLoaded`

- [module:event-store#event:ModifyTemplateField](#)
- [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);  
}
```

## Methods

### destroy()

Destroys the module.

---

*Documentation generated by JSDoc 3.3.3 on Mon Mar 13 2017 16:48:42 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: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.

**Properties**

Name	Type	Description
elem	HTMLElement	The element in which the module UI will be inserted.

Documentation generated by [JSDoc 3.3.3](#) on Mon Mar 13 2017 16:48:42 GMT-0400 (Eastern Daylight Time)

## 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.

```
(require("form-summary"))(viewer)
```

Creates the form summary module.

#### Parameters:

Name	Type	Description
viewer	Core	The core viewer to which the module will attach.

#### 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);  
}
```

#### Methods

`destroy()`

Destroys the module.

---

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 GMT-0400 (Eastern Daylight Time)

## 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.

### Properties

Name	Type	Description
elem	HTMLElement	The element in which the module UI will be inserted.

### Methods

Destroys the module.

---

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 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('myGlobalSetti

  });
}
```

`(require("global-settings-menu"))(viewer, options)`

Creates the global settings menu module.

#### **Parameters:**

viewer	Core	The core viewer to which the module will attach.
--------	------	--

options	Object	An options object.
---------	--------	--------------------

### 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 3.3.3* on Mon Mar 13 2017 16:48:42 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('myGlobalSetti

  });
}
```

```
(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**

Name	Type	Description
e1em	HTMLElement	The element in which the module UI will be inserted.

**Methods**

**destroy()**

Destroys the module.

---

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 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](#)

- [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 3.3.3](#) on Mon Mar 13 2017 16:48:42 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: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('myMultipleSel

    });
}
```

`(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**

Name	Type	Description
elem	HTMLElement	The element in which the module UI will be inserted.

Methods

Destroys the module.

---

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 GMT-0400 (Eastern Daylight Time)

## Module: notification

### Module: notification

Displays a notification.

```
(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.

Properties		
Name	Type	Description
elem	HTMLElement	The element in which the module UI will be inserted.

#### 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, {
```

```
document.getElementById('myPageNav')
    });
}
```

## Methods

### destroy()

Destroys the module.

---

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 GMT-0400 (Eastern Daylight Time)

## Module: page-navigation

### Module: page-navigation

Navigates pages.

#### Example

```
var PageNavigation = require('page-
navigation.js');

// a generic Viewer constructor
function Viewer(opts) {
    var myPageNav =
    PageNavigation(this, {
        elem:
    document.getElementById('myPageNav')
    });
}
```

`(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.

### 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 3.3.3* on Mon Mar 13 2017 16:48:42 GMT-0400 (Eastern Daylight Time)

## Module: profile-manager

### Module: profile-manager

Provides the ability to create and manage signatures.

#### 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('myProfileMana
});
```

```
(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**

Name	Type	Description
elem	HTMLElement	The element in which the module UI will be inserted.

## Methods

**destroy()**

Destroys the module.

---

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:42 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:**

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 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	Object	<optional>		A hash object used to store and access the metadata for each role in the form. The key for this hash object is the <code>formRoleId</code> of the form role.

**Properties**

Name	Type	Description
formRoleId	string	The id of the form role.
displayName	string	A friendly name for the form role.
fieldColor	string	The color to use for any field to which the form role is assigned, as a pound sign followed by a 6 character hexadecimal color code.
sortIndex	number	A number representing the sorting order of the form role.

**groups** Object <optional> A hash object used to store and access the metadata for each group in the form. The key for this hash object is the `groupId` of the group.

**Properties**

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. Possible values are: <ul style="list-style-type: none"> <li>• checkbox</li> </ul>
data	Object		Data associated with the group.

**Properties**

Name	Type	Description
multiple	boolean	Indicates whether or not the group allows multiple selections.

**readOnly** boolean <optional> Indicates whether this group is read only when signing

			the form.
		required	boolean <optional> Indicates whether this group is required when signing the form.
		formRoleId	string <optional> The form role id associated with the group.
fieldList	Object	A hash object used to store and access the metadata for each field in the form. The key for this hash object is the markId of the viewer mark.	
		<b>Properties</b>	
		<b>Name</b>	<b>Type</b> <b>Attributes</b> <b>Description</b>
		markId	number The viewer mark associated with the form field.
		fieldId	string A unique id for that field.
		displayName	string A friendly name for the field.
		template	string The data type of the field. Possible values are: <ul style="list-style-type: none"> <li>SignatureTemplate</li> <li>InitialsTemplate</li> <li>TextTemplate</li> <li>DateTemplate</li> <li>CheckboxTemplate</li> </ul>
		required	boolean Indicates whether this field is required when completing the form.
		readOnly	boolean Indicates whether this field is read-only when completing the form.
		horizontalAlignment	string Indicates the horizontal Alignment of text within the field Possible values are: <ul style="list-style-type: none"> <li>left</li> <li>center</li> <li>right</li> </ul>
		sortIndex	number A number representing the sorting order of the field, when displaying an ordered list.
		pageNumber	number The page number where the field is located.
		pageData	Object Represents metadata about the page at the time when the field rectangle was created or updated.
			<b>Properties</b>
		<b>Name</b>	<b>Type</b> <b>Description</b>
		width	number The width of the page.
		height	number The height of the page.
		rectangle	Object The location of the field on the document.
			<b>Properties</b>
		<b>Name</b>	<b>Type</b> <b>Description</b>
		x	number The x-coordinate of the top-left of the field in respect to the document.
		y	number The y-coordinate of the top-left of the field in respect to the document.
		width	number The width of the field.
		height	number The height of the field.
		fontName	string <optional> The font name to use for TextTemplate and DateTemplate fields.

fontColor	string	<optional>	The font color to use for TextTemplate and DateTemplate fields, as a pound sign followed by a 6 character hexadecimal color code.
multiline	boolean	<optional>	Indicates whether or not the field is multiline. This property is only used for TextTemplate fields.
characterLimit	number	<optional>	The amount of characters that can be entered in a TextTemplate field. It is a whole number greater than or equal to 0 that indicates the maximum number of characters allowed in a text field when completing the form, with 0 indicating that there is no limit.
formRoleId	string	<optional>	The id of the form role associated with the field.
groupId	string	<optional>	The id of the group that contains the field.
defaultValue	string   Object	<optional>	The default value of the field to be added on FormLoaded. If it is a TextTemplate, this value should be a string. If it is a DateTemplate, it should be an ISO formatted date string (e.g. 2016-11-16T18:23:08.030Z). The format will be dictated by the globalSettings.signatureDateFormat property. If it is a CheckboxTemplate, it may only be "" or "checked". If it is a signature, it should be an object with the following properties:

**Properties**

Name	Type	Attributes	Description
type	string	<optional>	The type of signature that will be placed in the field. Can be "text" for a text signature or "path" for a freehand signature.
value	string	<optional>	For a text signature, this is the actual text that will be in the signature. For a path signature, this is a subset of the SVG path standard, only including the M, L, and C commands (e.g. M0,0L1,1L1,0).
fontName	string	<optional>	The font to be used to render the signature. This only applies to text signatures and not path signatures.

value	string   Object	<optional>	<p>The value of the field. This property is only used in the <b>ESigner</b> and is undefined if the field has not been filled out. If it is a <b>TextTemplate</b>, this value should be a string. If it is a <b>DateTemplate</b>, it should be an ISO formatted date string (e.g. 2016-11-16T18:23:08.030Z). The format will be dictated by the <code>globalSettings.signatureDateFormat</code> property. If it is a <b>CheckboxTemplate</b>, it may only be "" or "checked". If it is a signature, it should be an object with the following properties:</p> <p><b>Properties</b></p> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Attributes</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>type</td> <td>string</td> <td>&lt;optional&gt;</td> <td>The type of signature that will be placed in the field. Can be "text" for a text signature or "path" for a freehand signature.</td> </tr> <tr> <td>value</td> <td>string</td> <td>&lt;optional&gt;</td> <td>For a text signature, this is the actual text that will be in the signature. For a path signature, this is a subset of the SVG path standard, only including the M, L, and C commands (e.g. M0,0L1,1L1,0).</td> </tr> <tr> <td>fontName</td> <td>string</td> <td>&lt;optional&gt;</td> <td>The font to be used to render the signature. This only applies to text signatures and not path signatures.</td> </tr> </tbody> </table>	Name	Type	Attributes	Description	type	string	<optional>	The type of signature that will be placed in the field. Can be "text" for a text signature or "path" for a freehand signature.	value	string	<optional>	For a text signature, this is the actual text that will be in the signature. For a path signature, this is a subset of the SVG path standard, only including the M, L, and C commands (e.g. M0,0L1,1L1,0).	fontName	string	<optional>	The font to be used to render the signature. This only applies to text signatures and not path signatures.
Name	Type	Attributes	Description																
type	string	<optional>	The type of signature that will be placed in the field. Can be "text" for a text signature or "path" for a freehand signature.																
value	string	<optional>	For a text signature, this is the actual text that will be in the signature. For a path signature, this is a subset of the SVG path standard, only including the M, L, and C commands (e.g. M0,0L1,1L1,0).																
fontName	string	<optional>	The font to be used to render the signature. This only applies to text signatures and not path signatures.																
globalSettings	Object		<p>Settings that apply globally to the template.</p> <p><b>Properties</b></p> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>signatureDateFormat</td> <td>external:"jQuery.fn"~DateFormat</td> <td>The format to use for dates when completing template fields.</td> </tr> </tbody> </table>	Name	Type	Description	signatureDateFormat	external:"jQuery.fn"~DateFormat	The format to use for dates when completing template fields.										
Name	Type	Description																	
signatureDateFormat	external:"jQuery.fn"~DateFormat	The format to use for dates when completing template fields.																	

**Example**

```
var fieldList = viewer.stateStore.getState('FieldList');
```

(inner) FormDefinition

**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 formRoles 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 formRoles object when a FormDefinition is loaded into the viewer.
groups	Array	<optional>		The data here is similar to the groups 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 groups object when a FormDefinition is loaded into the viewer.
formData	Array			The data here is similar to the fieldList 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 FieldList object when a FormDefinition 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> .
globalSettings	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
width	number	The width of the page.
height	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 3.3.3* on Mon Mar 13 2017 16:48:43 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:**

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
<code>dom</code>	<code>HTMLElement</code>	A parent DOM element, or jQuery-wrapped element, that contains the icons that need to be parsed.

---

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:43 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:FormLoaded`
- `module:event-store#event:FormCopied`
- `module:event-store#event:TemplateSaved`
- `module:event-store#event:TemplateSaveFailed`

### **Listens to Events:**

- `module:event-store#event:SaveTemplate`

## 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 3.3.3* on Mon Mar 13 2017 16:48:43 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:

viewer	Core	The core viewer to which the module will attach.
options	Object	An options object.
<b>Properties</b>		
<b>Name</b>	<b>Type</b>	<b>Description</b>
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('myTemplateMan

  });
}
```

Methods  
destroy()

Documentation generated by *JSDoc 3.3.3* on Mon Mar 13 2017 16:48:43 GMT-0400 (Eastern Daylight Time)

## Module: template-name-header

### Module: template-name-header

dfgsdf Provides the ability to display the currently loaded template name as a header.

```
(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 abject.

Properties		
Name	Type	Description
elem	HTMLElement	The element in which the module UI will be inserted.

#### 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:
```

```
});
```

## Methods

### **destroy()**

Destroys the module.

---

*Documentation generated by [JSDoc 3.3.3](#) on Mon Mar 13 2017 16:48:43 GMT-0400 (Eastern Daylight Time)*

## Module: text-input

### Module: text-input

A text input.

`(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

#### **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".-->
```

```
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);  
});
```

## 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()** → {HTMLInputElement}

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

## 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.

`(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		
Name	Type	Description
elem	HTMLElement	The element in which the module UI will be inserted.

#### Example

```
var ZoomFit = require('zoom-fit.js');

// a generic Viewer constructor
function Viewer(opts) {
    var myZoomFit = ZoomFit(this, {
        elem:
        document.getElementById('myZoomFit')
    });
}
```

#### Methods

`destroy()`

---

*Documentation generated by [JSDoc 3.3.3](#) on Mon Mar 13 2017 16:48:43 GMT-0400 (Eastern Daylight Time)*

## 1 Index

**.NET MVC 5, 169-173**

**.NET MVC 5 Sample, 135-141**

**.NET WebForms, 166-169**

**.NET WebForms Sample, 127-135**

**1 - Set up a Back-end , 61**

**2 - Install Viewer Assets & PAS , 94**

**3 - Integrate the Viewer with Your Application, 157**

**About PrizmDoc, 2-4**

**Accusoft Cloud-Hosted , 94-95 , 61**

**Accusoft Policy on Log Changes, 299**

**Add a Custom Button, 320-322**

**Add Features to the Template , 570-581**

**Add Keyboard Shortcuts, 322-326**

**Adding Custom Image Stamps, 235**

**Adjust Caching Parameters for PrizmDoc Server , 251-254**

**Adjust Office Conversion Settings for Optimal Performance, 254-256**

**Adjust Vector Conversion Settings for Optimal Performance, 257**

**Administrator Guide, 180-182**

**Affinity Tokens & Multi-Server Mode, 281-285**

**Annotate Documents, 476-477**

**Annotate Tab, 538-540**

**Annotation Layers , 351-352**

**API Data Types, 1012-1015**

**API Reference, 614**

**Architecture & Design, 308-311**

**Architecture Basics, 315-317**

**Attachments, 1077-1078**

**Back End Proxy, 1015-1017**

**Build a Custom User Interface, 326-330**

**Build the E-Signature Viewers, 434-435**

**Burning Annotated Content in the Viewer, 523-525**

**Burning Redacted Content in the Viewer, 525-527**

**Burning Signed Content in the Viewer, 527-529**

**Change Annotation Default Values, 330**

**Change Encryption Keys for Public use Token Generation, 257-259**

**Change the Position of the Menu Bar, 330-331**

**Check PrizmDoc Server Health, 92-94**

**Check the Connection to Accusoft Cloud-Hosted Services , 109**

**Check the Connection to your Self-Hosted Server, 125-126**

**Check the System's Health, 293-294**

**Class: AjaxResponse, 622-624**

**Class: BurnRequest, 624-634**

**Class: Comment, 634-646**

**Class: Conversation, 646-656**

**Class: ConversionRequest, 656-667**

**Class: DocumentHyperlink, 667-672**

**Class: Error, 672-673**

**Class: ESigner, 1186-1190**

**Class: Event, 673-676**

**Class: ImageStamps, 676-680**

**Class: LoadMarkupLayersRequest, 681-688**

**Class: Mark, 688-776**

**Class: MarkupLayer, 777-799**

**Class: MarkupLayerCollection, 799-807**

**Class: MouseTool, 807-816**

**Class: ObservableCollection, 816-820**

**Class: PrintRequest, 820-825**

**Class: Promise, 826-831**

**Class: SearchRequest, 831-839**

**Class: SearchResult, 839-848**

**Class: SearchTask, 848-852**

**Class: SearchTaskResult, 852-858**

**Class: SignatureControl, 863-865**

**Class: SignatureDisplay, 865-867**

**Class: TemplateDesigner, 1190-1194**

**Class: ThumbnailControl, 867-875**

**Class: Viewer, 875-877**

**Class: ViewerControl, 877-944**

**Command-Line Mode, 212-213 , 190-191**

**Configuration Options, 217-218**

**Configure a Cluster, 94**

**Configure Image Frame Rendering in the PDF Conversion Service, 259-260**

**Configure Log File Locations, 260-262**

**Configure Microsoft Office Conversion Connectivity, 262-264**

**Configure PAS in Your Server's Entry Point, 157-161**

**Configure PrizmDoc Application Services (PAS) , 235**

**Configure Server Connections, 124-125**

**Configure the E-Signature Viewers, 433-434**

**Configure the PrizmDoc Server , 240-243**

**Configure the Viewer, 217**

**Configuring Ports, 250**

**Configuring Skinny Comments Panel, 222-223**

**Connection Issues, 294-295**

**Contact Accusoft Support, 23**

**Content Conversion Demo, 393**

**Content Conversion Service, 1078-1086**

**Content Converters, 1017-1021**

**Content Converters Deprecated, 1022-1024**

**Convert Content with Content Conversion Service, 382-392**

**Copyright Information, 27**

**Create a Custom Mouse Tool, 331-332**

**Create a Custom Tab, 332-333**

**Create a Full Page Redaction, 497-499**

**Create a Polyline Annotation, 479-480**

**Create a Strikethrough Annotation, 480-481**

**Create a Text Hyperlink Annotation, 481-482**

**Create a Viewing Session, 161-162**

**Create Image Stamp Annotations, 477-479**

**Creators - Design a Template, 555**

**Customization Examples, 318-320**

**Customize Excel Document View Settings for PrizmDoc Server, 265-266**

**Customize Excel Pagination Settings for PrizmDoc Server , 264-265**

**Customize the E-Signature Viewers, 430**

**Customize the Markup, 333-335**

**Customize the Styles, 335-338**

**Customize the Viewer, 308**

**Defining the View Mode, 223-224**

**Definitions, 23-25**

**Deployment Licensing, 198**

**Deprecated Configuration Properties, 285-286**

**Design Basics, 311-313**

**Developer Guide, 304-305**

**Developer Reference, 1012 , 1077 , 1185-1186**

**Developer Reference , 620-621**

**Digital Rights Management Configuration, 224-225**

**Disable Excel Pagination for PrizmDoc Server , 267-268**

**Disable the Print Button, 338**

**Document Workflow, 19-21**

**Download Documents, 520**

**Downloading the Original Document, 520-523**

**Embed the Viewer, 162-166**

**Enabling Content Encryption, 225-229**

**End User Guides, 437**

**Error Reporting, 295**

**E-Sign Tab , 542-548**

**E-Signature Module, 550-551**

**E-Signing Overview, 551-555**

**Evaluation Licensing, 189-190**

**Exceeded Installation Limit , 194-196**

**External: jQuery, 621-622**

**External: jQuery.fn, 1194-1197**

**Feature Licensing, 213-214**

**Fill in Fields Programmatically, 435-436**

**Fill Out a Form, 591-602**

**Form Definitions, 1024-1028**

**Form Extractors, 1028-1034 , 1086-1093**

**Form Field Detector Error Messages, 586-588 , 302-303**

**Get an Evaluation License, 65-67**

**Get Started with PrizmDoc, 59-61**

**Getting Started, 295-299**

**Global, 622**

**Glossary, 23**

**Handle Specific Routes from PrizmDoc Application Services using Custom Logic , 356-359**

**Health, 1034-1035**

**Health Status, 1093-1097**

**How & when to use CORS with PrizmDoc, 239-240**

**How To, 381-382 , 446-447**

**How to Configure the Demo on Linux, 396-399**

**How to Configure the Demo on Windows, 393-396**

**How to Get an Evaluation License, 61-63**

**How to use Predefined Search , 229-233**

**HTML - General Help Topics**

Embed the Viewer, 162-166

Supported File Formats , 8-12

**HTML - How To - PrizmDoc Application Services (PAS)**

Configure PAS in Your Server's Entry Point, 157-161

Handle Specific Routes from PrizmDoc Application Services using Custom Logic , 356-359

Pre-Convert Documents, 359-362

PrizmDoc Application Services (PAS) Configuration, 236-239

Set up Your Database for use with PrizmDoc Application Services , 353-355

**HTML - How To - PrizmDoc Server**

Convert Content with Content Conversion Service, 382-392

Markup Burner XML Specification, 370-381

Set up a Viewing Session for a CAD Drawing which has XREF Dependencies , 407-415

**HTML Templates, 619-620**

**HTML5 Viewing , 1097-1104**

**Image Stamps, 1036-1037**

**Implement PrizmDoc Server Caching Strategies , 268-272**

**Initialization Parameters, 218-219**

**Install Asian Fonts, 86-87**

**Install on Linux, 80 , 107-109 , 122**

**Install on Windows, 67 , 96 , 111**

**Install PrizmDoc Server, 67**

**Install PrizmDoc Viewer, 111 , 95-96**

**Install the Viewer, 96-105 , 111-120**

**Integrate PrizmDoc Releases with Your Code, 318**

**Integrate Your Web Application with PAS, 166**

**Introduction, 305-307**

**JSP, 176-179**

**JSP Sample, 150-157**

**Keyboard Shortcuts, 548-549**

**Legacy Create Session, 1037-1039**

**Legal, 26-27**

**License Expired, 196-197**

**License Viewer (Deprecated) , 1104-1105**

**Licensing, 189**

**Linux, 275-276 , 272-274 , 292-293**

**Linux Installation, 82-86 , 122-124**

**Linux Requirements & Supported Environments, 80-82**

**Load Annotations, 494-496**

**Load Annotations from the Web Tier, 352-353**

**Local File Viewer Guide, 613**

**Localizing the Viewer, 233-234**

**Markup Burner, 1039-1041**

**Markup Burner XML Specification, 370-381**

**Markup Burners, 1105-1114**

**Markup Layers, 1041-1045**

**Markup XML, 1046-1047**

**Migrate from Accusoft Cloud-Hosted Servers to Self-Hosted Servers , 399-400**

**Migrate to PrizmDoc Application Services , 355-356**

**Mixin: Data, 944-948**

**Mixin: SessionData, 858-863**

**Modify viewer.js , 349-351**

**Module: button-set, 1197-1200**

**Module: checkbox-collection, 1200-1203**

**Module: date-picker, 1204-1205**  
**Module: download-signed-form, 1205-1206**  
**Module: download-signed-form-trigger, 1206-1208**  
**Module: dropdown, 1208-1211**  
**Module: event-store, 1211-1225**  
**Module: field-edit, 1225-1226**  
**Module: field-list, 1226-1227**  
**Module: fill-checklist, 1227-1228**  
**Module: fill-form-controller, 1228-1230**  
**Module: fill-main-toolbar, 1230-1231**  
**Module: fill-progress, 1231-1232**  
**Module: form-controller, 1232-1234**  
**Module: form-extraction, 1234-1235**  
**Module: form-summary, 1235-1236**  
**Module: form-tools, 1236-1238**  
**Module: global-settings-menu, 1238-1239**  
**Module: global-settings-trigger, 1239-1240**  
**Module: keyboard-controller, 1240-1241**  
**Module: multiple-selection, 1241-1243**  
**Module: notification, 1243-1244**  
**Module: page-navigation, 1244-1245**  
**Module: profile-manager, 1245-1246**  
**Module: state-store, 1246-1252**  
**Module: svg-icons, 1252-1253**  
**Module: template-io, 1253-1254**  
**Module: template-manager, 1254-1256**  
**Module: template-name-header, 1256-1257**  
**Module: text-input, 1257-1260**  
**Module: zoom-fit, 1260-1261**  
**Mouse Tools, 347-349**  
**Multi-Server Environments, 277-278**  
**Multi-Server Mode , 1114-1116**  
**Namespace: Ajax, 948-954**  
**Namespace: fn, 954-959**  
**Namespace: Language, 959-962**

- Namespace: MarkupLayerSchema, 979-980
- Namespace: MouseTools, 980-983
- Namespace: PCCViewer, 983-1000
- Namespace: Signatures, 1000-1001
- Namespace: Util, 1001-1011
- Natively Render Microsoft Office Documents, 400-401
- New Terms, 25-26
- Next Steps, 21
- No Internet Connection, 191-194
- Node-Locked, 198-204
- OEM, 207-209
- Optimize Cache Performance for Multi-Server Environments, 286-287
- Optimize Cache Performance for Multi-Server Mode, 280-281
- Overview, 215-217 , 438-446 , 614-619
- Overview of PrizmDoc, 7-8
- Package Log Files for Product Support, 301-302
- Perform Auto-Redaction, 401-405
- PHP, 173-176
- PHP Sample, 141-150
- Pre-Convert Documents, 359-362
- Pre-Loaded Search Parameters, 220-222
- Pre-Populate Fields in the E-Signature Viewer, 405-407
- Print Documents, 515-518
- Print Non-standard Size Documents, 518-520
- PrizmDoc Application Services, 187-189
- PrizmDoc Application Services , 353
- PrizmDoc Application Services (PAS), 16-18
- PrizmDoc Application Services (PAS) , 286
- PrizmDoc Application Services (PAS) Configuration, 236-239
- PrizmDoc Application Services (PAS) RESTful API
  - API Data Types, 1012-1015
  - Back End Proxy, 1015-1017
  - Content Converters, 1017-1021
  - Content Converters Deprecated, 1022-1024

Form Definitions, 1024-1028  
Form Extractors, 1028-1034  
Health, 1034-1035  
Image Stamps, 1036-1037  
Legacy Create Session, 1037-1039  
Markup Burner, 1039-1041  
Markup Layers, 1041-1045  
Markup XML, 1046-1047  
Search Tasks, 1047-1055  
Viewing Package Creators, 1055-1067  
Viewing Packages, 1067-1069  
Viewing Session, 1069-1076

## **PrizmDoc Application Services Database Administration & Maintenance, 239**

### **PrizmDoc E-Signature Viewer API**

Class: ESigner, 1186-1190  
Class: TemplateDesigner, 1190-1194  
Developer Reference, 1185-1186  
External: jQuery.fn, 1194-1197  
Module: button-set, 1197-1200  
Module: checkbox-collection, 1200-1203  
Module: data-persist, 1203-1204  
Module: date-picker, 1204-1205  
Module: download-signed-form, 1205-1206  
Module: download-signed-form-trigger, 1206-1208  
Module: dropdown, 1208-1211  
Module: event-store, 1211-1225  
Module: field-edit, 1225-1226  
Module: field-list, 1226-1227  
Module: fill-checklist, 1227-1228  
Module: fill-form-controller, 1228-1230  
Module: fill-main-toolbar, 1230-1231  
Module: fill-progress, 1231-1232  
Module: form-controller, 1232-1234  
Module: form-extraction, 1234-1235

Module: form-tools, 1236-1238  
Module: global-settings-menu, 1238-1239  
Module: global-settings-trigger, 1239-1240  
Module: keyboard-controller, 1240-1241  
Module: multiple-selection, 1241-1243  
Module: notification, 1243-1244  
Module: page-navigation, 1244-1245  
Module: profile-manager, 1245-1246  
Module: state-store, 1246-1252  
Module: svg-icons, 1252-1253  
Module: template-io, 1253-1254  
Module: template-manager, 1254-1256  
Module: template-name-header, 1256-1257  
Module: text-input, 1257-1260  
Module: zoom-fit, 1260-1261

## **PrizmDoc PDF, 1**

## **PrizmDoc Server, 184-187**

## **PrizmDoc Server , 278-280 , 365-366**

## **PrizmDoc Server (Accusoft Cloud-Hosted) , 19**

## **PrizmDoc Server (Self-Hosted), 18-19**

## **PrizmDoc Server RESTful API**

Attachments, 1077-1078  
Content Conversion Service, 1078-1086  
Developer Reference, 1077  
Form Extractors, 1086-1093  
Health Status, 1093-1097  
HTML5 Viewing , 1097-1104  
License Viewer (Deprecated) , 1104-1105  
Markup Burners, 1105-1114  
Multi-Server Mode , 1114-1116  
Redaction Creator, 1116-1122  
Search Contexts, 1122-1140  
Search Tasks, 1140-1151  
System Information (Deprecated), 1151-1155

Work Files , 1177-1185

## **PrizmDoc Viewer API**

Class: AjaxResponse, 622-624

Class: BurnRequest, 624-634

Class: Comment, 634-646

Class: Conversation, 646-656

Class: ConversionRequest, 656-667

Class: DocumentHyperlink, 667-672

Class: Error, 672-673

Class: Event, 673-676

Class: ImageStamps, 676-680

Class: LoadMarkupLayersRequest, 681-688

Class: Mark, 688-776

Class: MarkupLayer, 777-799

Class: MarkupLayerCollection, 799-807

Class: MouseTool, 807-816

Class: ObservableCollection, 816-820

Class: PrintRequest, 820-825

Class: Promise, 826-831

Class: SearchRequest, 831-839

Class: SearchResult, 839-848

Class: SearchTask, 848-852

Class: SearchTaskResult, 852-858

Class: SignatureControl, 863-865

Class: SignatureDisplay, 865-867

Class: ThumbnailControl, 867-875

Class: Viewer, 875-877

Class: ViewerControl, 877-944

Developer Reference , 620-621

External: jQuery, 621-622

Global, 622

HTML Templates, 619-620

Mixin: Data, 944-948

Mixin: SessionData, 858-863

Namespace: fn, 954-959  
Namespace: Language, 959-962  
Namespace: MarkSchema, 962-979  
Namespace: MarkupLayerSchema, 979-980  
Namespace: MouseTools, 980-983  
Namespace: PCCViewer, 983-1000  
Namespace: Signatures, 1000-1001  
Namespace: Util, 1001-1011  
Overview, 614-619

**Redact Documents, 497**

**Redact Search Results, 474-476**

**Redact Tab, 540-542**

**Redaction Creator, 1116-1122**

**Reference, 529-530**

**Release v12.0, 6-7**

**Release v12.1, 6**

**Release v12.2, 5-6**

**Release v12.3, 4-5**

**Reorganize Menus, 344-345**

**Responsive Layouts, 313-315**

**Run PrizmDoc Application Services on Multiple Servers, 287-288**

**Save Annotations, 496-497**

**Scroll the Viewer Programmatically, 338-340**

**Search Box and Icons, 462-463**

**Search Contexts, 1122-1140**

**Search Documents, 462**

**Search Patterns, Filters & Document Review Features, 463-466**

**Search Tab, 534-538**

**Search Tasks, 1047-1055 , 1140-1151**

**Search Tips, 299-301**

**Security Guidance, 182-184**

**Self-Hosted, 109-110 , 64-65**

**Server Hosting Options, 21-23**

**Set the Initial Zoom Factor, 345-346**

**Set up a Viewing Session for a CAD Drawing which has XREF Dependencies , 407-415**

**Set up Viewer Samples (Optional), 126-127**

**Set up Your Database for use with PrizmDoc Application Services , 353-355**

**Sign a Form Converted by the Form Field Detector, 608-613**

**Sign a Form with Multiple Roles, 602-608**

**Signers - Fill out a Form, 588**

**Sizing Servers, 184**

**Software License Agreement, 27-32**

**Start & Stop PrizmDoc Application Services, 272**

**Start & Stop PrizmDoc Server , 275**

**Subscribe to Events, 346-347**

**Substitute Fonts for Office Rendering Fidelity, 272**

**Supported File Formats , 8-12**

**System Configuration, 214-215**

**System Information (Deprecated), 1151-1155**

**System is Already Licensed for Evaluation, 197-198**

**System Requirements & Supported Environments, 110-111 , 95**

**Third-Party Attributions, 32-58**

**Tools, 530**

**Tools Reference, 555-563 , 588-591**

**Transfer Your Document to PrizmDoc Server , 415-418**

**Troubleshoot, 288**

**Troubleshoot Evaluation Licensing, 191**

**uiElements, 219**

**Unattended Install & Uninstall, 78-80 , 105-107 , 120-122**

**Understand the Core Components of PrizmDoc, 13-14**

**Uninstall PrizmDoc on Linux, 125**

**Uninstall PrizmDoc on Linux , 91-92**

**Uninstall PrizmDoc on Windows , 80**

**Upgrade to Central Configuration, 248-250**

**Upgrade Your Subscription, 63-64**

**Use a Viewing Session, 418-420**

**Use Annotation Comments, 503-506**

**Use Annotation Layers, 482**

**Use E-Signature Comments , 506-508**

**Use Redaction Reasons, 499-503**

**Use Skinny Comments, 511-515**

**Use Text Selection Options, 448-454**

**Use the Comment Feature, 503**

**Use the Fixed Search Terms Feature, 466-473**

**Use the Form Field Detector, 581-586**

**Use the PrizmDoc Server API , 366-370**

**Use the Proximity Search Feature, 473-474**

**Using a Custom Resource Path, 234-235**

**View Documents, 447-448**

**View Embedded Hyperlinks, 454-457**

**View Tab, 531-534**

**Viewer, 14-16**

**Viewer Guide, 437-438**

**Viewer Main Screen Options, 530-531**

**Viewer Modular Design, 430-433**

**Viewer Requirements, 12-13**

**Viewing Package Creators, 1055-1067**

**Viewing Packages, 362-365 , 1067-1069**

**Viewing Session, 1069-1076**

**Viewing Sessions, 1155-1177**

**Watermark Content in a Viewing Session, 420-430**

**Web Tier, 16**

**What's New?, 4**

**Windows, 276-277 , 274-275 , 288-292**

**Windows Installation, 69-78**

**Windows Requirements & Supported Environments, 67-69**

**Without an Internet Connection, 204-207**

**Work Effectively with Large Documents, 307-308**

**Work Files , 1177-1185**

**Work with Annotation Layers , 482-494**

**Work with Annotations Programmatically, 340-344**

**Work with Email Attachments, 460-462**

**Work with Sticky Mouse Tools, 549-550**

**Work with Thumbnails, 457-460**