Linedata Capitalstream

Capitalstream Document Server Cloud Guide

Document version: 07/14/2025

Linedata

Copyright notice

Table of Contents

About This Guide

Applicable Version

This guide applies to the following Document Server version: **Document Server Cloud**

Important version compatibility information

The following versions of Capitalstream are compatible with the corresponding versions of Document Server as listed below:

Capitalstream version	Compatible Document Server version	Available documentation	Image Location
Capitalstream Cloud (2024.12 onwards)	Document Server 2025.04.0.161	This document	cs-docker-releases/csdoc/v2025.04.0.161/
Capitalstream Cloud (2024.10 and below)	Document Server 2021.10.0.113	This document	cs-docker-releases/csdoc/v2021.10.0.113/

Version Information

Document Server Cloud is a continuation of the Document Server 6 series compatible with Capitalstream Cloud releases ONLY. Please note, the Document Server Cloud is NOT compatible with non-cloud releases of Capitalstream.

Please note, document server version **2025.04.0.161** is the current version of the Document Server Cloud series that is compatible with Capitalstream Cloud (2024.12 onwards).

Configuration And Setup

For information about how to configure and maintain the document generation feature in the Capitalstream application, please refer to the *Capitalstream Administrator's Guide*.

For the Capitalstream Cloud version, installation of the Document Server requires assistance from Linedata. Please contact your Linedata Representative for more information.

Document Server Cloud Feature Overview

Capitalstream Document Server Cloud is the server software that accepts requests from users who want to generate, convert, or merge Capitalstream documents in Microsoft Word, Adobe PDF, or Microsoft Excel formats. This capability enables financial organizations to use Microsoft Word and Adobe PDF documents as well as Microsoft Excel spreadsheets for their underwriting and documentation functions.

The following sections provide further details about the features available in Document Server Cloud:

Feature Overview

Document Server 6 continues to support all features present in earlier versions of the Document Server, including the following functionality:

Microsoft Word Integration – Enables financial institutions to use existing Microsoft Word document templates for loan documentation and internal uses such as credit write-ups. Includes the ability to export data automatically from Capitalstream to a Word document.

Document Conversion – Converts Excel and Word documents to Adobe PDF format. Provides both security and flexibility by only allowing authorized users to generate documents in various file formats.

Document Merging – Can merge multiple documents into a single, aggregated document. Combined with conversion, this feature enables the construction of complex master documents from disparate individual file sources (such as a loan write-up that includes both narrative credit summaries and spreadsheet-style financial analysis). This feature also makes it easy to prepare a documentation set for electronic transfer to a customer or authorized third party.

Excel Documents – Enables two-way communication that supports downloading of data from Capitalstream into Excel spreadsheet templates and uploading of data from Excel spreadsheets back into Capitalstream, which provides users the ability to work offline.

Automatic Generation of Documents – Administrators can configure a Capitalstream workflow to generate documents or document lists automatically.

Custom Clauses – Supports Document Manual Clauses, which are reusable paragraphs of text that can be inserted into a template file or added to documents on an *ad hoc* basis. Clauses are often used to include special legal statements that are used across document types. Once you have set up doc clauses, the functionality for inserting clauses into a document exists via the **Add Document** button on any Documents tab.

Configurable Security – Makes all documents subject to security restrictions applied by the Capitalstream administrator. Depending on how document options for a particular entity is configured, a user might be able to upload, download, and e-mail documents, generate documents from templates, and merge documents together into a document package. Additionally, administrators can configure document checklists to help manage the documentation tasks related to a specific transaction or customer.

Private Label Document Branding – Provides a means for company-specific branding on documentation generated for a given transaction. If more than one organization is involved in a transaction, documentation can be generated that bears the logo of any of the involved organizations.

Easy Access – Provides users access to document functionality via the **Documents** tab, which is available on many tab sets throughout Capitalstream, including the **Customer** tab set, the **Finance Request** tab set, and the **Transaction** tab set.

Support for Rich Text Editing (RTE) formatted fields – Supports the Rich Text Editing (RTE) feature (which enables users to add text and paragraph formatting, images, and tables to RTE-enabled fields) by applying such formatting to content merged into a Capitalstream-generated document. If a text area field contains text formatted with or images inserted by the rich text editor, these formatting attributes or images can be automatically applied to a document that contains that field when you generate that document as Word or PDF in Capitalstream ---provided that an appropriate document tag has been inserted in the document template. (Please refer to the RTE implementation documentation for details on how to set up this feature.)

End User Features

Capitalstream Document Server provides end users with a variety of features for working with documents. Users can:

- Generate documents
- View documents
- Merge documents
- E-mail a document or group of documents
- Add from a list of available documents
- Create custom documents
- Upload new or existing documents from their computer
- Access details, history, and notes for a specific document

What's new in Document Server 6

This section describes new and enhanced features available in this release of Document Server.

> **Note**
> Generally speaking, if you are upgrading from an older version of Capitalstream that used an earlier version of the Document Server, end-users of Capitalstream will not see any changes to the current document processing and management capabilities of Capitalstream. However, from an administrator's perspective, there are some table formatting limitations for documents generated using Document Server 6 detailed later in this document.

Support for templates created in Microsoft Office

Document Server 6 now supports more recent versions of the Microsoft Word templates. For details on the supported Microsoft releases, refer to the release history in this document.

Upgrades to server architecture

Document Server 6 upgrades the server architecture to provide the benefits detailed in the following sections.

A more robust, platform-independent document generation architecture

Document Server 6 changes the way Word documents are created by Capitalstream, eliminating the server-side use of Microsoft Word for generating Word documents with merged Capitalstream data. This architecture change creates a more stable server environment and reduces dependence on Microsoft Office licenses.

For the most recent detailed information about supported Microsoft Office document formats, please refer to the *Linedata Capitalstream 20.01 Client Hardware and Software Requirements* document available in the standard Capitalstream documentation package.

Simpler server configuration and maintenance

Document Server 6 provides easier configuration and maintenance because of an improved software architecture in which Word documents can now be parsed and the value of document fields can be resolved solely by the Capitalstream application server – thus eliminating the need to call back to a web

service on the Document Server machine to resolve document fields. This change should make server configuration and troubleshooting a simpler process than in previous versions of the Document Server.

Table formatting limitations in generated documents

The following includes a listing of the table formatting limitations for documented generated using Document Server 6:

- Row alignment is misaligned for caption rows. Note that this misalignment will not be noticeable if the background is white
- PDF conversion does not convert the table border color to the value specified in the document table settings in Capitalstream; the border color is always output as black. This is only an issue with PDF generation. Generated Word .docx files use the table border color specified in the document table settings
- The vertical alignment of generated tables defaults to centered
- Header row text is always black
- PDF conversion auto-wraps text if a column is not wide enough
- One blank line will appear before tables generated by CS_TBL or CS_RTETBL document fields
- Labels (static text) preceding a CS_TBL or CS_RTETBL table will appear underneath the generated table
- Manual drawing of tables using HR tags in the doc table scripts in Capitalstream is not recommended, the recommended approach is to use the HTML table style to build the tables

Document Server Cloud Release History

In this section

Current version: Document Server 2025.04.0.161

Document Server 2025.04.0.161 is the current release for Capitalstream Cloud and it is the current version of the Document Server Cloud series.

Important!
For the Capitalstream Cloud version, installation of the Document Server requires assistance from Linedata. Please contact your Linedata Representative for more information.

Changes made in Document Server 2025.04.0.161

- DOCE-91 - DocServer enhancement to handle image resizing efficiently
- DOCE-89 - Fix issues with double border coming up in documents for table generated through a script
- DOCE-92 - Fix doc generation errors due to invalid characters in RTE fields
- DOCE-93 - Docfield resolved value logging dynamically on demand or in case of error

Certified Office Version
Microsoft® Word for Microsoft 365 MSO (Version 2406 Build 16.0.17726.20078) 64-bit

Changes made in Document Server 2024.12.0.159

- Docx4J libraries upgraded to the latest stable version 8.3.11.
- Addressed the following issues in generated documents -
    - Inconsistent sizing for the rows (especially the enlarged last row) of a table inserted in RTE fields.
    - Table borders not coming up for tables copied from web/excel into RTE fields.
    - Incorrect alignment for indented text.
    - Text highlighting not reflecting.
    - Images extending off margins.
- New properties added to handle table row height alignments for Document Tables and Document Fields content and left margin alignment for RTE fields content paragraph tags.
- Updates to existing properties to adjust alignment with Document Server upgrade.
- Updates to Document Tables configuration to retain the table structure.
- New properties to define the default font family and font size for text in RTE fields that is entered and saved without any explicit selection of font style and font size.
- Updated/Renewed the Aspose license in the Document Server with the 2025 version.

Changes made in Document Server 2024.07.0.147

- MEI-898 - Documents are not generating when using updated existing document templates or new document templates due to Microsoft Office updates. A new property has been added to Cloud Admin Console to configure any future missing xml namespace.

> **Certified Office Version**
> Microsoft® Word for Microsoft 365 MSO (Version 2406 Build 16.0.17726.20078) 64-bit

Changes made in Document Server 2024.05.0.144

- HF20X-157 - Documents are not generating when using updated existing document templates or new document templates due to Microsoft Office updates.

Changes made in Document Server 2023.12.0.137

- MEI-812 - Updated/Renewed the Aspose license in the Document Server with the 2024 version

Changes made in Document Server 2022.09.0.121

- MEI-423 - Upgraded log4j libraries to resolve RCE 0-day exploit

Changes made in Document Server 2021.12.0.119

- DOC-19 - Alignment issues in the generated document on using
 tag
- DOC-20 - Content before HTML table doesn't get rendered on the generated document on using CS_RTE tag
- DOC-22 - Issues with styling on the generated document
- DOC-26 - Updated/Renewed the Aspose license in the Document Server with the 2022 version
- DOC-28 - Removed the Document Server dependency on the Cloud Admin Console

> **Note**
> The changes carried out for the Document Server 2021.10.0.113 were rolled into Capitalstream Release 21.11 which resulted in a change in the build image version from 2021.10.0.113 to 2021.11.0.117. Please note that no updates were made to the Document Server 2021.11.0.117.

Changes made in Document Server 2021.10.0.113

- DOC-8 - Character Code (& #160;) is displayed in the generated document
- DOC-9 - Numbering issue in the generated document for a specific content
- DOC-10 - Issue with font size on the generated document
- DOC-11 - Issue with <li> on generated document
- DOC-12 - Content before HTML table doesn't get rendered on the generated document
- DOC-13 - Font Styles aren't applied correctly
- DOC-14 - Extra space observed between Table heading and RTE table content
- DOC-15 - Spacing between paragraphs are lost for RTE Fields
- DOC-16 - Formatting issues when multiple tables are rendered using document table
- DOC-17 - Font Style is getting applied to all the text inside the font tag
- DOC-18 - Document Bookmark is not rendering in the specified font type and size after upgrading the document

Note

As part of this Doc server release, below are the known issues/limitations:

- Only the following types of numbering lists are supported – bullets, alphabets, numbers, and Roman numbers
- The numbered items will be indented as per the default word behavior
- Table cell spacing will work only in Word and not in pdf
- Unnecessary paragraph tags will create empty lines in the generated document
- if the line break is required then the text has to be enclosed in the paragraph tag

Hotfix HF_2021_09_01 needs to be deployed before installing Document Server 2021.10.0.113

Changes made in Document Server 2021.05.0.87

- DOC-1 - RTE image size in a generated doc is larger than the image in the CS RTE field
- DOC-2 - Unable to merge pdf's with password protection enabled
- DOC-3 - Text Format [bold and underline] pasted on CS RTE fields do not render in the generated document
- DOC-4 - Error is displayed during document generation when document template is created in Office 365

- DOC-5 - Contents in Signers/Co-Borrowers/Guarantor table sections are getting truncated in certain Model document
- DOC-6 - Observed Font size and Font type differences on the generated document with PDF format

Note

As part of this Doc server release:

- The Doc Server has been certified on the Word version of Microsoft 365 and 2016 plus - Version 2008, 2103
- In addition to the previously included fonts, the liberation font has been added
- Merge functionality for PDFs only work for the following protection types:
  - Copy Protected
  - Mod (Write) Protected
  - Print Protected
- When one or more PDFs with the above protection types are merged, the resulting merged PDF will not be protected
- Merge functionality does not work for 'Open Protected' PDFs

Changes made in Document Server 6.0.8

Document Server 6.0.8 fixes the following issue:

- Bug 25274 - Resized pictures in an RTE field are generated in the original size in documents

Changes made in Document Server 6.0.7

Document Server 6.0.7 fixes the following issues:

- Bug 24939 - Font color not being recognized when in a footer
- Bug 24903 - Support of old doc formats for PDF merge

Note

The fix for Bug 24939 has revealed the following technical limitation within the docx4j open source code: when there are HTML styles defined in a Document Field ("doc tag") in the header or footer, you cannot include a table within the Document Field itself. If you need a table in the header or footer, you must define it in the template itself and not within the Document Field. You can, however, put a Document Field within a table defined in the template, and it will render correctly.

Changes made in Document Server 6.0.6

Document Server 6.0.6 consisted of fixes for the following issues:

- Bug 24887 - Upgrade Aspose .jar files to latest version
- Bug 24888 - Temporary files are not deleted after documents are generated
- Bug 24940 - Document Server outage issue with large size loop

Note

The fix for Bug 24940 revealed the following issue in the docx4j open source code: when there are nested HTML styles – such as bold, underline, and italics – some styles are lost if the HTML tags are ordered in a certain way. Specifically, if italic tags are used prior to underline tags for the same text, then the underline style will be not appear. The solution is to switch the order of the tags to avoid this issue. The correct order should be bold first, then italic, and finally underline. Any combination should use that order, that is, bold then underline. Note as well that this issue would mainly impact document field scripts containing formatting.

Sample:

<html><body>This is bold textThis is bold/italic text<u>This is bold/italic/underline text</u></body></html>

renders as the following:

This is bold text*This is bold/italic text*<u>*This is bold/italic/underline text*</u>

Changes made in Document Server 6.0.5

Document Server 6.0.5 consisted of fixes for the following issues:

- Bug 24843 - Word style of character positioning (extra spaces) not supported

- Bug 24844 - Paragraph alignment is not maintained consistently
- Bug 24845 - Hidden characters in a field can cause document generation to fail

To fix Bug 24845, a new property, **nonPrintableCharSet**, was added that is set to exclude 28 of the current set of 31 non-printable ASCII characters. This property is in the Document Server .WAR file in the ValidateSourceForDocxSupportConf.properties file located in the \WEB-INF\classes\com\capitalstream\docserver\spring\ folder. The three non-printable characters that are NOT excluded are:

- tab
- linefeed
- carriage return

When a non-printable character is removed during document generation, it is usually replaced with a blank space in the generated document. However, in certain cases, a blank space may not be inserted, so it is recommended that when copying and pasting from certain sources that might contain non-printable characters (such as PDF files), you should review the generated document to ensure that it displays as expected.

Changes made in Document Server 6.0.4

Document Server 6.0.4 consisted of fixes for the following issues:

- Bug 24744 - Bookmarks within nested tables are not processed by the Document Server
- Bug 24809 - Label preceding CS_RTE HTML does not appear next to CS_RTE HTML in a Microsoft Word table

In addition, the fix for Bug 24809 resolved the CS_RTE field blank space and blank line issues in the partial fix for Bug 24914 in Document Server 6.0.3.

Changes made in Document Server 6.0.3

Document Server 6.0.3 included an additional formatting file, docx4j-ImportXHTML, located in the Document Server war file in the \WEB-INF\classes folder. It enables the user to set a default font for unspecified fonts in dynamic content, i.e., document fields. See Appendix 5 of this document for more information.

In addition, document server 6.0.3 consisted of fixes for the following issues:

- Bug 24370 - Text is always being left-justified 6.0.4
- Bug 24371 - Font point size is one point smaller for generated 6.0.4

- Bug 24665 - Static text preceding a CS_RTE field is missing from the generated document 6.0.4
- Bug 24701 - Document headers are not handled properly, sometimes they are not resolved in the generated document 6.0.4

The fix for bug 24194 "DS 6 - Extra blank lines before tables and RTE fields in DS 6.0.2" has been reverted due to the fix causing another issue. A new, partial fix for the original issue in Bug 24194 has been made. The new behavior is described in Appendix 1 below.

Changes made in Document Server 6.0.2

Document Server 6.0.2 was an internal-only release and included an upgrade to the Axis .JAR files to address a memory leak in the third party library, a modification to the DDACallback timeout with respect to the .JAR file change, a timing change (when document XML data is obtained) associated with the automated document formatting feature, and two bug fixes related to document formatting.

- Bug 24140 - Create new version of Document Server to upgrade Axis ,JAR files and fix related issue with DDACallback timeout and a timing change for the automated document formatting feature
- Bug 24194 - Extra blank lines before tables and RTE fields
- Bug 24195 - Leading spaces added in a particular doc field scenario

Changes made in Document Server 6.0.1

Document Server 6.0.1 added functionality to control font selection, font size, and spacing globally for generated Word documents.

- Bug 24138 - Create new version of Document Server to implement automation of font, font size, and spacing specification

Changes made in Document Server 6.0

Document Server 6.0 implemented support for Word templates created in Microsoft Office 2016 and saved in .docx format.

In addition, the following issues that occurred in Document Server 5 have been fixed in Document Server 6.0:

- Bug 23993 - Default font parameter in applicationcontext.xml is not functioning
- Bug 23946 - Boldface text in document fields is not as expected
- Bug 23940 - Rich Text Editing - Initial space not handled properly in resulting HTML
- Bug 23689 - Update Aspose license files for 2019

Document Server Technical Information

Important version and compatibility information

- Document Server Cloud is **required** for use with Capitalstream Cloud. Earlier versions of Document Server can NOT be used with Capitalstream Cloud.

Document Server is a Java-based web service that generates documents in Microsoft Word and Microsoft Excel formats for use in Capitalstream during the loan origination and booking process. It supports the following document formats:

- .docx
- .xlsx
- .xlxm

The server can also convert these document formats into Adobe PDF files for use in the Capitalstream system.

Important upgrade note

Legacy file formats such as .DOC, .XML, and, .XLS are no longer supported in Document Server 6. If you are upgrading from a previous version, Capitalstream 10.8 and above includes a built-in utility that you can use to automatically convert all existing legacy file formats (such as .DOC, .XML, and, .XLS) to the newer file type supported by Document Server 6. This conversion process is NOT automatic, and the documents MUST be manually converted in Capitalstream by an administrator before you can use Document Server 6. For more information on performing this conversion in Capitalstream Administration, see the "Configuring Capitalstream Settings" section of this document.

Warning

Please note, Document Server Cloud table formatting is based on HTML5 and inline CSS standards. Because there are differences between formatting supported with HTML5 that is now used and HTML4 used by the prior version of the Document Server, it is recommended that implementation teams review document template formats and adjust them as needed.

Additional technical information

In the previous version of Document Server, Microsoft Word documents were parsed on the Document Server machine to obtain a list of document fields. These document fields were then resolved into values by calling a web service on the Capitalstream application server. This DDA callback URL was used to help the Document Server locate and invoke the web service on the Capitalstream application server. The web service call was necessary in older versions of Capitalstream because the Capitalstream system did not have the ability to resolve document field values on its own. The disadvantage of requiring direct, two-way communication between the Document Server and the Capitalstream application server was that configuration of the communication between the two servers could be complicated and difficult to troubleshoot – especially if the Document Server machine and the Capitalstream application server machine were located on different domains.

Capitalstream 10.8 and above in tandem with Document 6 has improved this design. A Word document can now be parsed and the value of document fields can be resolved solely by the Capitalstream application server. This architecture eliminates the need to call back the web service on the application server to resolve document fields.Note that if for performance reasons, you want document fields to be pre-processed on the Document Server machine rather than being processed on the Capitalstream application server machine, you can enable pre-processing of document fields to occur on the Document Server machine before documents are sent to the Capitalstream application server. This feature, however, is turned off by default and must be enabled. For information about how to do this, see the "Configuring Capitalstream Settings" section of this document.

Configuring Capitalstream Settings

Configuring Capitalstream settings

Perform the following steps in Capitalstream administration to enable certain Document Server 6 features.

Enable pre-processing of document fields

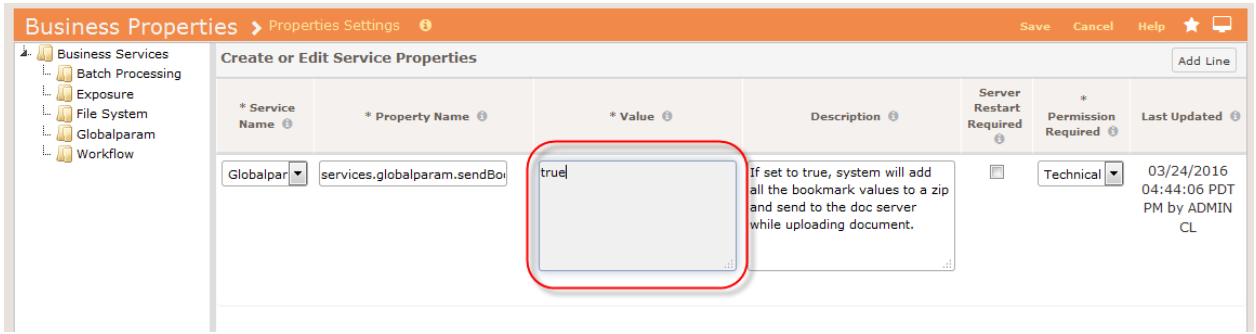
In the previous version of Document Server, Microsoft Word documents were parsed on the Document Server machine to obtain a list of document fields. These document fields were then resolved into values by calling a web service on the Capitalstream application server. This DDA callback URL was used to help the Document Server locate and invoke the web service on the Capitalstream application server. The web service call was necessary in older versions of Capitalstream because the Capitalstream system did not have the ability to resolve document field values on its own. The disadvantage of requiring direct, two-way

communication between the Document Server and the Capitalstream application server was that configuration of the communication between the two servers could be complicated and difficult to troubleshoot – especially if the Document Server machine and the Capitalstream application server machine were located on different domains.

Capitalstream 10.8 and above in tandem with Document 6 has improved this design. A Word document can now be parsed and the value of document fields can be resolved solely by the Capitalstream application server. This architecture eliminates the need to call back the web service on the application server to resolve document fields.

If for performance reasons you want document fields to be pre-processed on the Document Server machine rather than being processed on the Capitalstream application server machine, you can enable pre-processing of document fields to occur on the Document Server machine before documents are sent to the Capitalstream application server. This feature, however, is turned off by default and must be enabled. Follow these steps below to enable this feature:

1. Navigate to Properties Config: **My Tools** > **System Setup** > **System Config** > **Properties Config**.
2. Use the search box in the screen header bar to find the property named **services.globalparam.sendBookmarkValuesToDocServer**.
3. Click **Edit**.
4. Enter **true** for the **Value** field.



5. Click **Save**.

The feature is now enabled

Enable HTML5 font style mapping for table text

For the Document Server to properly map fonts set up to be used in document tables to the same fonts in the generated document, follow these steps:

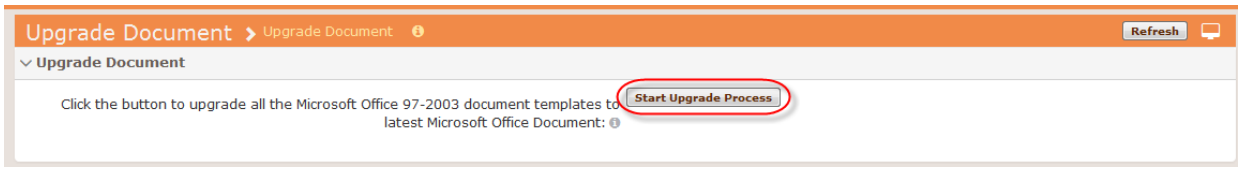
1. Navigate to Properties Config: **My Tools** > **System Setup** > **System Config** > **Properties Config**.

2. Use the search box in the screen header bar to find the property named **services.globalparam.isDocServer5Configured**.
3. Click **Edit**.
4. Enter **1** for the **Value** field.

5. Click **Save**.

Upgrade legacy documents to Document Server 6 formats (upgrading users only)

1. In Capitalstream administration, navigate to My Tools > Feature Setup > Document Setup > Upgrade Document.
2. Click Start Upgrade Process.



3. When the upgrade has completed, a message will appear informing you that the conversion process has completed successfully.

> Important - Once the document templates have been upgraded there is not a way to convert them back to .xml.

Appendix 1: Table Formatting In Documents Generated Using Document Server Cloud

In this section

Because Document Server 6 uses the newer HTML5 standard rather than the HTML4 standard used in the previous version of the Document Server, certain formatting differences may occur in table formatting for documents generated using Document Server 6. This appendix describes some of the known formatting differences to assist you when designing document tables for use with Document Server 6.

Summary of known table formatting issues

Known issues in Document Server 6

- Row alignment is misaligned for caption rows. Note that this misalignment will not be noticeable if the background is white.
- PDF conversion does not convert the table border color to the value specified in the document table settings in Capitalstream; the border color is always output as black. This is only an issue with PDF generation. Generated Word .docx files use the table border color specified in the document table settings.
- The vertical alignment of generated tables defaults to centered.
- Header row text is always black.

- PDF conversion auto-wraps text if a column is not wide enough.
- One blank line will appear before tables generated by CS_TBL or CS_RTETBL document fields.
- Labels (static text) preceding a CS_TBL or CS_RTETBL table will appear underneath the generated table.
- Manual drawing of tables using HR tags in the doc table scripts in Capitalstream is not recommended, the recommended approach is to use the HTML table style to build the tables.

Known issues in Document Server 4

For comparison purposes, the following list summarizes the know issues with table formatting in document tables generated using Document Server 4.

- The table text font size cannot be changed; it is always 11 point in the generated Word table even if a different font size is indicated for the document table in Capitalstream settings.
- Table rows have a taller height than the rows in a Document Server 5 generated table and contain extra padding.
- The vertical alignment of table text defaults to top alignment.
- Document generation crashes if the template uses a fixed-column table.

Example 1: Differences in generated Word files between versions 4 and 6

Below are two tables generated with exactly the same document table settings and template file. The only different is that Document Server 6 uses the .docx file format for templates while Document Server 4 uses an the .xml file format as a template.

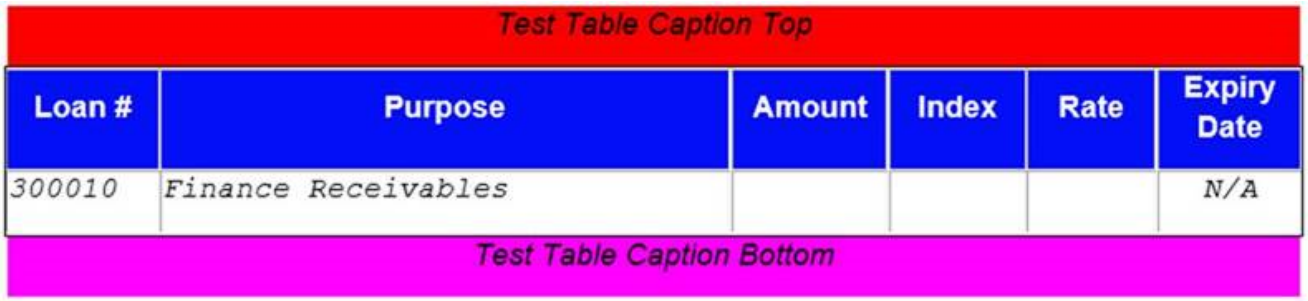
Document Server 6 generated table (.docx file)

Test Table Caption Top

Loan #	Purpose	Amount	Index	Rate	Expiry Date
300010	Finance Receivables				N/A

Test Table Caption Bottom

Document Server 4 generated table (.docx file)



Test Table Caption Top

Loan #	Purpose	Amount	Index	Rate	Expiry Date
300010	Finance Receivables				N/A

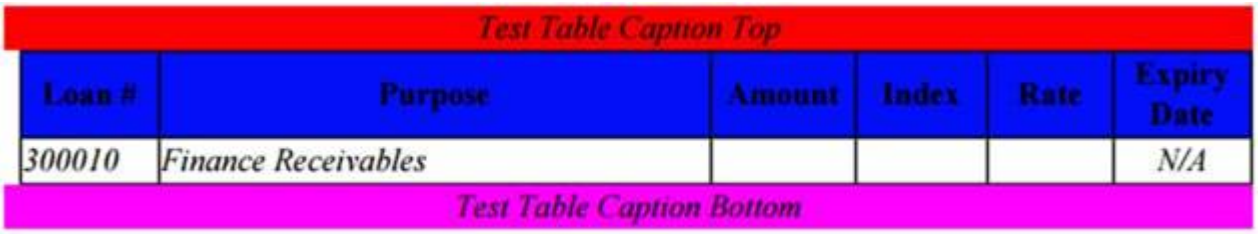
Test Table Caption Bottom

The following table summarizes the differences between the two Word document tables generated with the same settings and template but using different versions of the Document Server:

Formatting difference	Comment / Possible workaround
The Document Server 4 generated table row has a taller height.	There is no setting for row height in Capitalstream document table setup, and so there is no workaround for this issue.
The Document Server 4 generated table font is 11 points while the Document Server 6 font size is 12 points.	The font setting for the document table text in the template used is 12 points, so the Document Server 6 document correctly maps the font size. The anomaly between the two font sizes is attributable to a known issue with Document Server 4.
The Document Server 4 generated table font is Arial while the Document Server 6 font is Times New Roman.	The font setting for the document table text in the template used is Arial. This issue can be fixed by ensuring that the global parameter **services.globalparam.isDocServer5Configured** has been enabled. See the section "Configuring Document Server" earlier in this guide for more information about how to enable this global parameter
The Document Server 4 header row font color is white while the Document Server 6 header font color is black.	There is no setting for header row font color in Capitalstream document table setup. The default font color is black in Microsoft Word.
The grid line is thicker in the Document Server 6 generated table.	There is no setting for grid line thickness in Capitalstream document table setup
The Document Server 4 generated table aligns text to the top of the row; the Document Server 6 generated table aligns text to the center of the row.	The alignment setting for the document table text in the template used is centered, so the Document Server 6 document correctly maps this setting to the generated document. The anomaly between the two font sizes is attributable to a known issue with Document Server 4.

Example 2: Differences in generated .PDF files

Document Server 6 generated table (.pdf file)



Test Table Caption Top					
Loan #	Purpose	Amount	Index	Rate	Expiry Date
300010	Finance Receivables				N/A
Test Table Caption Bottom					

Document Server 4 generated table (.pdf file)

Test Table Caption Top					
Loan #	Purpose	Amount	Index	Rate	Expiry Date
300010	Finance Receivables				N/A
Test Table Caption Bottom					

The following difference exists between the two PDF document tables generated with the same settings and template but using different versions of the Document Server:

- The table generated using Document Server 6 has alignment issues with the top caption and bottom caption; the table generated using Document Server 4 also has alignment issues with the same elements, but the issue is not as noticeable.

Appendix 2: Supported Highlight Colors For Rich Text Fields In Document Server Cloud

Document Server 6 only supports a list of 16 colors (see below) to highlight colored text taken from rich text formatted fields Capitalstream. This behavior is due to the limited set of predefined highlight colors supported by the docx4j Java component. If an unsupported color is used, the Document Server processing engine ignores the color, and no highlight color is applied to the text.

The following table lists the supported colors:

Color name	Hex code
black	000000
blue	0000FF
cyan	00FFFF
green	008000
magenta	FF00FF
red	FF0000
yellow	FFFF00
white	FFFFFF
darkBlue	00008B
darkCyan	008B8B
darkGreen	006400
darkMagenta	8B008B
darkRed	8B0000
darkYellow	FFD700
darkGray	A9A9A9
lightGray	D3D3D3

Appendix 3: Supported Fonts In Document Server Cloud

In this section

Supported fonts

Document Server 6 uses the docx4j Java library to generate documents in Microsoft Word format. Fonts supported for use with Document Server generated documents will differ depending on the operating system on which the Document Server is installed. If Document Server is installed on the Microsoft Windows operating system, then the fonts included out-of-the-box with Windows will be available for use; if Document Server is installed on the Linux operating system, then the fonts included out-of-the-box with Linux will be available for use.

Important Note!
Any additional fonts not included out-of-the-box with the operating system MUST be obtained separately by the financial institution. Note that commonly used fonts that are included out-of-the-box with Microsoft Windows are NOT included with the Linux operating system and must be licensed separately for a fee.

More information
For more information about how fonts work in the docx4j Java library, please refer to the docx4j documentation available at http://www.docx4java.org.

Viewing a list of supported fonts in Microsoft Windows

The complete list of fonts supported by the docx4j Java component in Windows can be viewed in XML format in the MicrosoftFonts.xml file located on the Document Server Windows machine. To view the MicrosoftFonts.xml file, follow these steps:

1. Copy and unzip the `<docserver_install_dir>/WEB-INF/lib/docx4j-3.3.x.jar` into a folder.
2. Locate the MicrosoftFonts.xml file in the extracted folder in the `/org/docx4j/fonts/Microsoft/` folder.
3. Open the file in a text editor.

Adding more fonts to Capitalstream

To add more fonts for use in Capitalstream document generation, use the following procedure:

1. Ensure that the desired font is installed on the Document Server machine. The font installation procedure will differ depending on what operating system is installed on the Document Server machine. Please refer to the relevant documentation for your operating system for more information about installing new fonts. Note as well that the desired font must also be supported for use by the docx4j Java component (see previous section for details).
2. Log in to Capitalstream as a user who has administrator privileges.
3. Navigate to **My Tools** > **System Setup** > **Lookup Setup** > **User List Lookups.**
4. Select **Font List** from the **Lookup** dropdown menu.
5. Click **Edit**.
6. Click **Add Line**.
7. Enter the code and description for the font from the list in the MicrosoftFonts.xml file.
8. Click **Save**.
9. Navigate to **My Tools** > **System Setup** > **Lookup Setup** > **Reset Lookups**.
10. Click **Reset**.
11. Verify that the font works by including it in a document table and then generating the document, downloading it, and opening it:
    a. Navigate to **My Tools** > **Feature Setup** > **Document Setup** > **Document Tables**.
    b. Search for and select a table. A good one that is easy to set up and is found in the model database is **DeliveryandAcceptance**.
    c. Click **Edit**.
    d. Set the **Table Font Type** and **Table Font Size**.
    e. Click **Save**.

f. Create a document template with a **CS_TBL** tag referencing the table – for example, **CS_TBL_DeliveryandAcceptance**.
g. Create a document template instance (**My Tools** > **Feature Setup** > **Document Setup** > **Document Templates**) on an entity. For the example table, use the **FINANCIAL_TRX** entity.
h. Generate the document on the entity. For the example table, create a finance request and transaction for any customer and include one or more collateral items. Then generate the document on the transaction.
i. Download the document and observe the font is being used for data within the table.

Installing bar code fonts

If you intend to use bar codes in your documents, you first need to install the required bar code fonts on the Document Server machine.

> **Note**
> You will need to obtain these fonts first. Many third-party font vendors offer bar code fonts for purchase if your organization does not already have bar code fonts for use.

The bar code font installation procedure will differ depending on what operating system is installed on the Document Server machine. Please refer to the relevant documentation for your operating system for more information about installing fonts.

After the bar code font is installed, refer to the instructions in the previous section for how to add new fonts for use in document generation in Capitalstream.

Troubleshooting font issues

- If your fonts are not appearing properly in your generated documents (for example, in the document table you created to verify that you had set up a new font properly), check the following ares:
  - Ensure that the font is properly installed on the Document Server machine.
  - Ensure that the font is supported for use by the docx4j Java component. Refer to the MicrosoftFonts.xml file for a full list of supported fonts.
  - Ensure that the font has been configured properly in Capitalstream so that the font lookup code matches EXACTLY (including case-sensitivity) what is in the MicrosoftFonts.xml file.

Appendix 4: Automatic Document Font Formatting

In older versions of Document Server (version 5.03 and earlier), if content in a generated document had not been formatted with a font style in the document template, a default font from Microsoft Word would be used to display the content instead. When this font substitution occurred, it caused font mismatches within the document – that is, two different fonts would be unintentionally mixed together in the content.

To address this font formatting mismatch issue, an Automatic Document Formatting feature has been added to this version of Document Server. The feature applies a font style, font size, and line spacing changes to an entire document automatically after that document has been generated by the Document Server and all bookmarks have been resolved with data.

Configuration

All formatters are defined and configured in the applicationContext.xml control file for the Document Server.

The following lines define the font style to apply to the entire document by the font formatter after the document has been generated:

```
<!--  the font to use for the Word ML formatter-->
<bean id="default_font" class="java.lang.String">
   <constructor-arg value="Arial Narrow"/>
</bean>
```

However, there are fonts that should not be changed in a generated document, such as barcode fonts, so these fonts should be excluded from being reformatted. The following lines define the fonts to exclude from being changed by the font formatter. The following sample codes tells the formatter not to change the barcode fonts in the document:

```
<!--  list of fonts to exclude from change for the Word ML font formatter  -->
<util:list id="exclude_fonts" value-type="java.lang.String">
  <value>Barcode39</value>
  <value>IDAutomation</value>
  <value>IDAutomationSC128L</value>
  <value>IDAutomationSC128M</value>
  <value>IDAutomationSC128S</value>
  <value>IDAutomationSC128XL</value>
  <value>IDAutomationSC128XS</value>
  <value>IDAutomationHC39M</value>
</util:list>
```

The values for the following properties define the font formatter's behavior:

- **font** - font style to use for formatting (set in default_font, see above)
- **fontSize** - font size (in points) to use formatting
- **overwriteFont** - if text is already formatted with an existing font, this determines whether to overwrite the existing font with the new font.
- **overwriteFontSize** - if text is already formatted with an existing font size, this determines whether to overwrite the existing font with the new font size.

The following code contains sample settings:

```
<!--
Word ML formatter to update the font and the font size of a document.
font: font name (set in default_cont, see above)
font size: in pt, ex: 8, 9, 10, etc...
overwriteFont: flag to overwrite existing font
overwriteFontSize: flag to overwrite existing font size
excludeFonts: fonts to exclude for change (set in exclude_fonts, see above)
 -->
<bean id="wordMLFontFormatter"
class="com.capitalstream.docserver.command.word.generate.WordMLFontFormatter">
       <property name="font" ref="default_font" />
       <property name="fontSize" value="12" />
       <property name="overwriteFont" value="true" />
       <property name="overwriteFontSize" value="true" />
<property name="excludeFonts" ref="exclude_fonts" />
</bean>
```

The values for the following properties define the line spacing formatter's behavior:

- **beforeSpacing** - this sets the "Before" spacing in the Word Paragraph format
- **afterSpacing** - this sets the "After" spacing in the Word Paragraph format.

- **linespacing** - this sets the "Line Spacing" in the Word Paragraph format: **1** for single-spacing, **1.5** for 1.5 line spacing, and **2** for double-spacing.

The following code contains sample settings:

```
<!--
Word ML format to update the line spacing.
bforeSpacing: Before spacing, in pt, ex: 0, 1, 2, 3, etc...
afterSpacing: After spacing, in pt, ex: 0, 1, 2, 3, etc..
lineSpacing: Line spacing, ex: 1, 1.5, 2
 -->
<bean id="wordMLLineSpacingFormatter"
class="com.capitalstream.docserver.command.word.generate.WordMLLineSpacingFormatter" >
        <property name="beforeSpacing" value="0" />
        <property name="afterSpacing" value="0" />
        <property name="lineSpacing" value="1" />
</bean>
</util:list>
```

The following lines define the main class which runs all Word ML formatters. The formatter is disabled by default. It can be enabled by commenting out the **wordMLFormatters** property, as shown in the following sample:

```
<bean id="wordUniversalFormatter"
class="com.capitalstream.docserver.command.word.generate.WordUniversalFormatter">
    <!-- uncomment following line to enable the formatter -->
        <!-- <property name="wordMLFormatters" ref="wordMLFormatters" /> -->
</bean>
```

Limitations

- The formatter only works with unprotected documents.
- The formatter does not check if the configured font style is available and supported in the system.
- The maximum font size allowed is based on the Word limit (1600).
- The formatter does not apply font changes to the following elements in a document:
    - Headers
    - Footers
    - Charts
    - Graphs
    - Images

- The formatter does not overwrite or remove macros in the document.
- The formatter only applies to generated Word (DDA) documents. It does not apply to uploaded documents (Quick Upload) or to Excel and PDF documents.

Notes

- When it overwrites the fonts in the template, the formatter replaces the font and font size for all formatted text including text in the **CS_FLD**, **CS_RTE**, **CS_TBL**, **CS_CLS**, and **CS_BAR** doc tags.
- Since **CS_BAR** data are ordinarily being overwritten and thus not appearing as actual barcodes, an exclusion formatter has been added so that these fonts can be excluded from being overwritten. Examples of excluded values added to the Document Server's applicationContext.xml control file include various barcode fonts.
- If the overwrite functionality is set to "false" for either the font or font size, then formatted text will not be re-formatted (overwritten). However, if the text is not specifically formatted – that is, the text is merely entered without highlighting and selecting a font/font size -- the text will be reformatted.
- Bold, italics, and underline formatting are retained even when the font and font size are changed.
- PDF conversion of generated Word documents works normally, with the overwritten fonts being "passed through" during the PDF conversion process.

Examples

Sample 1

Font and font size overwritten:

- Default font: Comic Sans MS
- Default font size: 16
- Overwrite font: true
- Overwrite font size: true

Sample of content in a generated Word document with these settings with Automatic Document Formatting enabled:

3 Customer Name Test: **MJB Test Company**

Asset Table:

Quantity	Asset Description (including make, model and serial number)	Supplier

Sample of the same content in a generated Word document with Automatic Document Formatting disabled:

3 Customer Name Test: **MJB Test Company**

Asset Table:

Quantity	Asset Description (including make, model and serial number)	Supplier

Word template (with Doc Tags) from which the document has been generated:

3 Customer Name Test: **##CS_FLD_CustomerCompanyName##**

Asset Table: ##CS_TBL_DeliveryandAcceptance## *

*Although the table doc tag is in Times New Roman, the table header formatting is in Arial.

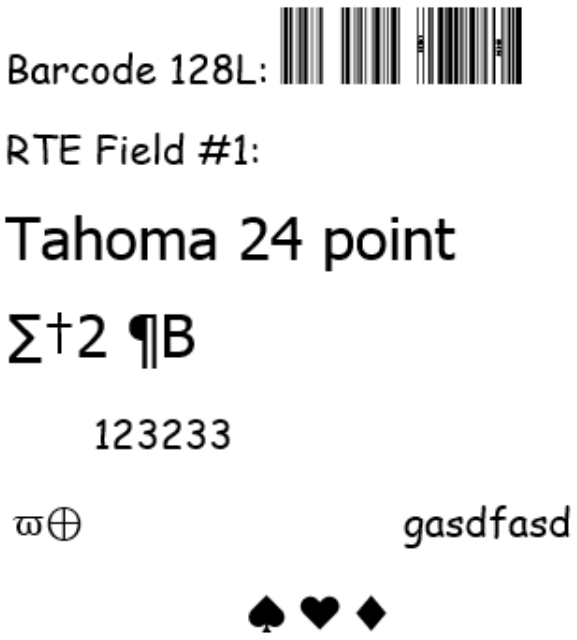
Sample 2

Exclusion of barcode fonts and another font (Tahoma):

- Default font: Comic Sans MS
- Default font size: 16
- Overwrite font: true
- Overwrite font size: true
- Excluded fonts:

```
<util:list id="exclude_fonts" value-type="java.lang.String">
         <value>Barcode39</value>
         <value>IDAutomation</value>
         <value>IDAutomationSC128L</value>
         <value>IDAutomationSC128M</value>
         <value>IDAutomationSC128S</value>
         <value>IDAutomationSC128XL</value>
         <value>IDAutomationSC128XS</value>
         <value>IDAutomationHC39M</value>
         <value>Tahoma</value>
</util:list>
```

Sample of content in a generated Word document with these settings with Automatic Document Formatting enabled:



Sample of the same content in a generated Word document with Automatic Document Formatting disabled:

Barcode 128L: 

RTE Field #1:

Tahoma 24 point

Σ†2 ¶B

123233

ϖ⊕ gasdfasd

♠♥♦

Word template (with Doc Tags) from which the document has been generated:

Barcode 128L: ##CS_BAR_128L##

RTE Field #1:

##CS_RTE_CUSmemo##

Appendix 5: Using The Docx4j-ImportXHTML Formatting File To Set A Default Font

The formatting file, docx4j-ImportXHTML.properties, is located in the Document Server WAR file in the \WEB-INF\classes folder. The file contains several parameters that can be configured, but only the following parameter, shown below with the its default value, is supported for use with the Capitalstream Document Server:

```
docx4j-ImportXHTML.fonts.default.serif=Times New Roman
```

When this font value is set, dynamic content that is not explicitly formatted with a specific font will be generated in this font. That is, the value of this parameter sets the default font that will be used for dynamic content in generated documents if no font is specifically set for that content.

> **Important note on use of HTML in document fields**
> When HTML is used in document fields ("doc tags"), these fields should be expressed in the document template as CS_RTE fields – and not as CS_FLD fields. CS_FLD document fields that contain HTML content may not appear completely or accurately in the generated. document.

Appendix 6: Docx4j Version Attribute

Certain Microsoft Windows regional settings may cause issues with Microsoft Word being able to open documents generated by the docserver. The particular region which was investigated and a solution found was French (France). A configuration change within the docsever will correct this issue:

1. Install the Docserver per standard installation documentation.
2. On the Docserver, navigate to the <Docserver_install_dir>\WEB-INF\classes\docx4j.properties file.
3. Find the docx4j.App.write attribute and set it to "false". docx4j.App.write=false
4. Restart the Docserver.

A selection from the docx4j.properties file showing the attribute set to "false" is below:

```
# These will be injected into docProps/app.xml
# if App.Write=true
docx4j.App.write=false
docx4j.Application=docx4j
docx4j.AppVersion=3.1.0
# of the form XX.YYYY where X and Y represent numerical values
# WARNING: -SNAPSHOT will cause Word 2010 x64 to treat the docx as corrupt!
```

> **Note**
> After this change is made, if preexisting generated documents contain the docx4j version attribute, users will have to regenerate the document to create a document without the attribute. There isn't a way to remove the attribute after the doc is generated – which is why a new version of the document must be generated.

Appendix 7: RTE Uploaded Image Resize In Document

In older versions of Document Server, images uploaded in the RTE fields do not render as expected in the generated document. The images above a certain height and width do not fit in the screen of the generated document.

To address this image overflow issue, new Doc server automatically resizes the height and width of the image if it is beyond the set threshold ratio/value. This threshold value is configurable in ValidateSourceForDocxSupportConf.xml. Details and functionality of each attribute introduced in the xml has been provided in configuration.

Configuration

The following attributes are introduced under styleAttributesMap which are part of ValidateSourceForDocxSupportConf.xml.

- maxImageWidth
- maxImageHeight
- imageUnit
- imageWidthResize
- imageHeightResize

```
<property name="styleAttributesMap">
<!-- This  map is being used to replace any style name which is not supported by docx4j-->
<!-- The key is the style name which is not supported and value is the replacement to be used for such styles -->
    <map>
        <entry key="WindowText" value="black" />
        <entry key="WindowFrame" value="black" />
        <entry key="Window" value="" />
        <entry key="padding" value="margin" />
        <entry key="maxImageWidth" value="400" />
        <entry key="maxImageHeight" value="445" />
        <entry key="imageUnit" value="px" />
        <entry key="imageWidthResize" value="0.75" />
        <entry key="imageHeightResize" value="0.75" />
    </map>
</property>
```

maxImageWidth & maxImageHeight:

- Default value - maxImageHeight:445, maxImageWidth:400
- While generating a document, if the image width or height uploaded in the RTE field is above the threshold value then image resize will happen based on the default values provided in order to avoid image overflow in the generated document

imageUnit:

- Default value - px, Represents unit type

imageWidthResize:

- Default value - 0.75
- If the image width is under the threshold value then it will use this attribute to resize image on generated document
- If the image size in the generated document is different compared to the RTE uploaded image then we can use this attribute to adjust the size of the image in the generated document

imageHeightResize:

- Default value - 0.75
- If the image height is under the threshold value then it will use this attribute to resize image on generated document
- If the image size in the generated document is different compared to the RTE uploaded image then we can use this attribute to adjust the size of the image in the generated document

Appendix 8: Merge Password Protected PDF

In older versions of Document Server, If a user tries to merge a password protected pdf, the merge fails. Apache Pdf box api was used in the older versions of the Doc Server which do not support merging of a password protected pdf.

To address this issue, itextPDF api has been used in the latest Doc Server which support's merging of Password protected pdf.

Configuration

ApplicationContext.xml:

```
<bean class="com.capitalstream.docserver.command.pdf.merge.MergePdfCommand">
    <property name="name" value="SimpleMerge()" />
    <property name="MergePdfProcessor" ref="MergePdfProcessorItext" />
</bean>

<bean id="MergePdfProcessorPdfBox"
   class="com.capitalstream.docserver.command.pdf.merge.MergePdfProcessorPdfBox">
</bean>
<bean id="MergePdfProcessorItext"
   class="com.capitalstream.docserver.command.pdf.merge.MergePdfProcessorItext">
</bean>
```

Please note that user's can specify the implementation type [either Apache PDFBox or ITextPDF] in applicationcontext.xml. User can either opt to use the pdfbox api implementation which was used in the prior versions or opt for itextpdf implementation which has been introduced as part of DS 2021.05.0.87.

By Default, itextPDF implementation will be used.

Appendix 9: Using Font Identifier

Font Identifier

In older versions of Document Server, when an external font is not specified for a document bookmark, then it resolves the value using the default configured font. But this approach was causing issues wherein the static and dynamic content was being displayed using different fonts. When we highlight the document bookmark and choose a different font for it then that is considered as an external font.

To resolve the issue, the latest document server has been updated to support multiple **Font Identifiers.**

Font Identifier	Description
0	**Universal Font -** Use configured font name and size universally across the complete generated document including static and dynamic content
1	**Global Default for Dynamic Content** - Use configured font and size as default font & size for dynamic content only if the external font is not specified. Static content font and size will be based on template default.
2	**Template Default for Dynamic Content** - Use default font and size from the template as default font & size for dynamic content only if the external font is not specified. Static content font and size will be based on template default.

Font identifier to be used can be configured by specifying a valid value (0,1 or 2) for the following configuration property - **docserver.applicationcontext.defaultFontIdentifier** in the Cloud Admin Console**.** The default value is 2.

Font Styling Precedence

For a document field type of **CS_FLD,** font styling can be applied to content in two ways - through document field script or by using the controls in Microsoft Word by highlighting the document bookmark and choosing a font type, size, etc. So if styling is applied within Microsoft word, then it is considered as external and is given precedence over the styling specified as part of the document field script. For example, for a document bookmark if the external font type is chosen as **"Times New Roman"** and in the document field script if the font type is specified as **"Arial"** as part of the font tag, then the external font **"Times New Roman"** will be applied.

Configuration Properties

Following properties have been added/updated as part of Document Server 2021.10.0.113.

Property	Default Value	Description
docserver.HtmlAutoConvertConf.fontSize Multiplier	15, 20, 24, 27, 36, 48, 72	Font size multiplier values for content rendered using font tag.
docserver.applicationcontext.defaultFont Identifier	2	
docserver.applicationcontext.fontSize	24	Default font size to be used for rendering content. It works in conjunction with the default font identifier configurations. For Font Identifier - 1, the size in which the content needs to be rendered should be multiplied by 2 and specified. For example, to render the content in font size 11, then the value should be specified as **22.** For Font Identifier - 0, the size in which the content needs to be rendered should be specified as is. For example, to render the content in font size 11, then the value should be specified as **11.**
docserver.applicationcontext.defaultFont	Arial Narrow	Default font to be used for rendering content. It works in conjunction with the default font identifier configurations.
docserver.document.table.rowheight	40	Table Row Height for Document tables content
docserver.document.field.rowheight	20	Table Row Height for Document fields content
docserver.HtmlAutoConvertConf.paraVisit orBean.marginTop	0in	Margin top configuration for paragraph tag
docserver.HtmlAutoConvertConf.paraVisit orBean.marginBottom	8pt	Margin bottom configuration for paragraph tag
docserver.HtmlAutoConvertConf.paraVisit orBean.marginLeft	-6pt	Margin left configuration for paragraph tag
docserver.HtmlAutoConvertConf.tableVisi torBean.marginLeft	-9px	Margin left configuration for table tag

Property	Default Value	Description
docserver.HtmlAutoConvertConf.liVisitorBean.marginTop	0in	Margin top configuration for list tag
docserver.HtmlAutoConvertConf.liVisitorBean.marginBottom	8pt	Margin bottom configuration for list tag
docserver.HtmlAutoConvertConf.listParentNodeVisitorBean.marginLeft	0px	Margin left configuration for list parent tag (OL, UL)

Config property "docserver.applicationcontext.enableWordMLFormatters" has been deprecated and same functionality can be achieved by changing the config property "docserver.applicationcontext.defaultFontIdentifier" as 0