Docmosis v3.3.0 Release Notes

Jan 2015

New Features

<table>
<tr><th>#</th><th>Change</th></tr>
<tr><td>1</td><td>
New HTML-injection feature. The new tag:

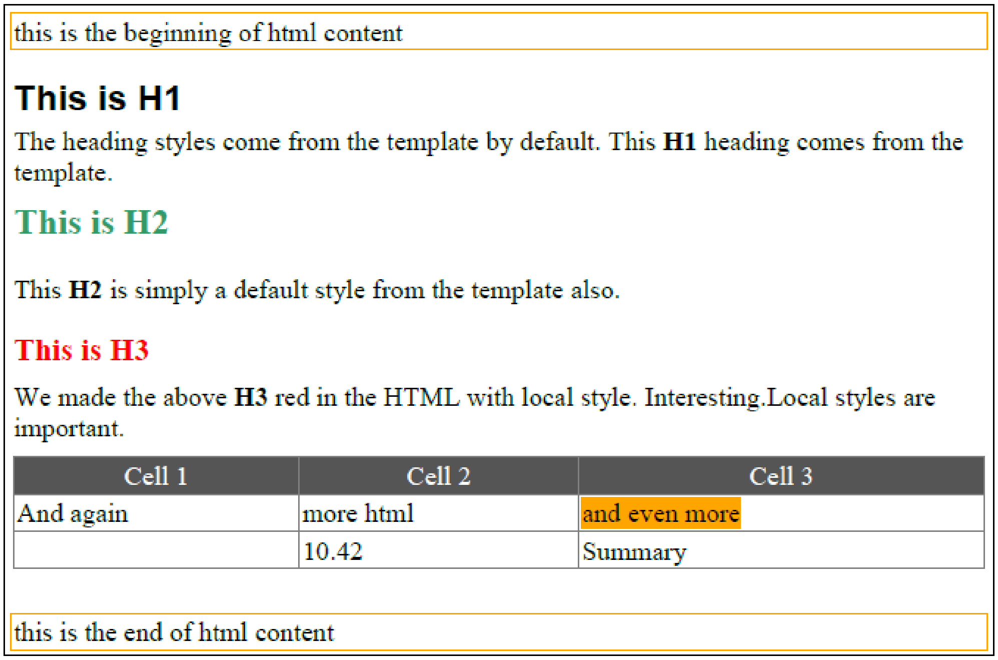
```
 <<html:myData>>
```

will process the given data as html and render the html into the document.

So, the following HTML:

```
<p style="border:1px solid orange; width:100%">this is the beginning of html content</p>
<h1>This is H1</h1>
The heading styles come from the template by default.  This <b>H1</b> heading comes from the
template.
<h2>This is H2</h2>
This <b>H2</b> is simply a default style from the template also.
<h3 style="color:red">This is H3</h3>
<p style="width:100%">
We made the above <b>H3</b> red in the HTML with local style.  Interesting.
Local styles are important.
</p>
<table width="100%"><tr>
<td style="text-align:center;border:1px solid gray;background-color:#555555"><span
style="color:white">Cell 1</span></td>
<td style="text-align:center;border:1px solid gray;background-color:#555555"><span
style="color:white">Cell 2</span></td>
<td style="text-align:center;border:1px solid gray;background-color:#555555"><span
style="color:white">Cell 3</span></td>
</tr><tr>
<td style="border:1px solid gray">And again</td>
<td style="border:1px solid gray">more html</td>
<td style="border:1px solid gray"><span style="background-color:orange">and even more</span></td>
</tr><tr>
<td style="border:1px solid gray">&nbsp;</td>
<td style="border:1px solid gray">10.42</td>
<td style="border:1px solid gray">Summary</td>
</tr></table>
<p>&nbsp;</p>
<p style="border:1px solid orange; width:100%">this is the end of html content</p>
```

Will render into a <<html:myData>> field like this:
</td></tr>
</table>

<table>
<tr><th>#</th><th>Change</th></tr>
<tr><td></td><td>
this is the beginning of html content
<h1>This is H1</h1>
The heading styles come from the template by default. This <b>H1</b> heading comes from the template.
<h2>This is H2</h2>
This <b>H2</b> is simply a default style from the template also.
<h3>This is H3</h3>
We made the above <b>H3</b> red in the HTML with local style. Interesting.Local styles are important.
<table>
<tr><th>Cell 1</th><th>Cell 2</th><th>Cell 3</th></tr>
<tr><td>And again</td><td>more html</td><td>and even more</td></tr>
<tr><td></td><td>10.42</td><td>Summary</td></tr>
</table>
this is the end of html content</td></tr>
<tr><td>2</td><td>Image Alt Text can now be dynamically changed. Adding a Docmosis tag to the alt-text of an image will cause document processing to process the tag and update the alt-text.</td></tr>
<tr><td>3</td><td>Accessibility updates for PDF output. To allow PDF documents to be more useful to low-vision users. The API call:
<code>ConversionInstruction.setPdfTagged(boolean)</code>
causes the PDF result to have extra information (such as alt-text for images) which assist accessibility-tools to present the PDF document to the user.</td></tr>
<tr><td>4</td><td>Improved document handling of document fields –bookmark cross-references will be updated dynamically if they contain dynamic content.</td></tr>
</table>

API Changes

The following API changes should be noted

Class / Interface	Change
`ConversionInstruction`	New method `setPdfTagged(boolean)` to add extra information to the PDF output for low-vision assistance.

Bug Fixes / Technical Changes

#	Change
1	Fixed a bug where multiple adjacent set-variable fields were causing an error in template analysis.
2	Fixed a bug where template processing wasn’t allowing multiple nested conditional sections on a single line with a Docmosis set-variable field.
3	Improved processing to stop blank lines being left in document after Docmosis content is stripped.

Docmosis v3.2.0 Release Notes

Sep 2014

New Features

<table>
<tr><th>#</th><th>Change</th></tr>
<tr><td>1</td><td>Improvements to the Java download:
- Library loading and class loading
- Confirfguration and property setting
to allow docmosis embedded converters to be used by multiple applications deployed into same Web or JEE container.

Particularly the new Configuration class allows properties to be set/overridden programmatically.</td></tr>
<tr><td>2</td><td>New “step down” feature to allow templates to traverse data in steps and “down” first instead of across first. This means data can be written down columns as well as across rows. This applies to both repeating sections (eg “<<rs_items:step3down>>”) and to table rows (eg “<<rr:items:step3down>>”).

For example, given a list/array of items
[a,b,c,d,e,f,g]
the directive
<<rr_items:step3Down>>
inside a table will create 3 columns of data and populate the first column top to bottom, then move to the second and third columns:
<table>
<tr><td>a</td><td>d</td><td>g</td></tr>
<tr><td>b</td><td>e</td><td></td></tr>
<tr><td>c</td><td>f</td><td></td></tr>
</table>
Using the same data, the directive:
<<rr_items:step2Down>>
inside a table will create 2 columns of data and populate the first column top to bottom, then move to the second:
<table>
<tr><td>a</td><td>e</td></tr>
<tr><td>b</td><td>f</td></tr>
<tr><td>c</td><td>g</td></tr>
<tr><td>d</td><td></td></tr>
</table>
Docmosis will automatically balance the number of rows to fit the given data into the desired number of columns.

As with the “stepN” directive, the “stepNdown” directive allocates variables names $i1, $i2 … to allow you to reference data in your array. For a “step3down” directive, $i1 will be the item for column1, $i2 for column2 and $i3 for column3. In a table, the “items” data can be mapped to a 3 column down-first table as follows:</td></tr>
</table>

<table>
<tr><th>#</th><th>Change</th></tr>
<tr><td></td><td>
<table>
<tr><td colspan="3"><<rr_items:step3Down>></td></tr>
<tr><td><<$i1>></td><td><<$i2>></td><td><<$i3>></td></tr>
<tr><td colspan="3"><<er_>></td></tr>
</table>

Note that $i1, $i2 and $i3 are automatically created by Docmosis to reach the items to be placed into the first, second and third column of the current row. The “items” data may be simple data or they could be structured objects. If they are simple objects, the above template example will render them as expected. If the “items” list contains structured objects, for example person data, then the following shows how the name of the person can be referenced:

<table>
<tr><td colspan="3"><<rr_items:step3Down>></td></tr>
<tr><td><<$i1.name>></td><td><<$i2.name>></td><td><<$i3.name>></td></tr>
<tr><td colspan="3"><<er_>></td></tr>
</table>
</td></tr>
<tr><td>3</td><td>There is a new tag <<noRowColoring>> (aka <<noRowColouring>>) which disables the automatic row colouring feature of Docmosis when expanding repeating rows in a table. (sometimes table-row colouring is not desirable).
The <<noRowColoring>> tag can appear in a table to disable it for that table. The tag can also appear in the text body of the document which will disable the row colouring in all following tables.</td></tr>
<tr><td>4</td><td>Dates, Booleans and Numeric data passed in as textual information (such as with XML or JSON data) can now be re-formatted by the template. This has been achieved by extending the renderers to also be able to parse data items and even according to specified formats.

For example, dates:
  <<myDate{renderer=date(‘dd/MMM/yyyy’)}>> will render myDate into dd/MMM/yyyy
And if your date data is in a special format, you can tell Docmosis how to parse it with a second parameter:
  <<myDate{renderer=date(‘dd/MMM/yyyy’,’EEE MMM dd HH:mm:ss zzz yyyy’)}>>

And booleans:
  <<myItem{renderer=boolean(‘yn’)}>> will render myItem into y or n values

And numbers:
  <<myVal{renderer=number(‘$00.00’)}>> will render myVal into $00.00 style values.

Please see the template guide for more information about the features of the renderers.</td></tr>
<tr><td>5</td><td>Java API and Command Line raw Conversion is now provided by a new API Class: DocumentConverter. This new class does not perform data population based on a template, it is simply to be used to convert between formats (eg ODT->PDF, or DOC->PDF). The DocumentConverter can convert documents at scale using the built in features of Docmosis for scaling document production.
A command-line version is provided as shown by example7 in the Docmosis-Java download.

Please see the API for more information.</td></tr>
</table>

#	Change
6	Two new built-in variables have been created: $rownum $rowidx These items present the current row number (or row index which starts from zero) when repeating data. These new variables are handy when using the "step" functions which affect the $itemnum and $idx variables.

API Changes

The following API changes should be noted

Class / Interface	Change
`Configuration`	New class allowing properties to be set programmatically.
`SystemManager`	A new method: `initialise(Configuration configuration)` has been added to allow Docmosis to be initialized with programmatic configuration overrides.
`DocumentConverter`	New class providing direct format conversion rather than document production. Example7 in the Java download contains a command line example of converting files to multiple formats
`DateRenderer`	The built in date renderer has been extended to parse dates from text data allowing text data to be interpreted and reformatted. Various date formats are applied by default.
`BooleanRenderer`	The built in boolean renderer has been extended to parse boolean from text data allowing text data to be interpreted and reformatted
`NumberRenderer`	New built-in renderer that can reformat numeric data, even when that data is provided in text format.

Bug Fixes / Technical Changes

#	Change
1	Fixed issue where unusual configuration could result in infinite loops in data streaming
2	Fixed issue where occasionally a plain text field might not be recognized as a field in a docX template.
3	Fixed issue whereby document-merging was not working with the latest "fresh" build of Libre Office (4.3.0.4). Libre Office is correcting the bug but we corrected for it anyway.

Docmosis v3.1.0 Release Notes

May 2014

New Features

#	Change
1	**Reverse engineering support from Templates** The `TemplateStructureExtractor` and `TemplateStructureProcessor` classes have been extended to allow sample data to be created for a given template. The list of fields, repeating sections, images etc in a template can be obtained in a descriptive form for programs to use. Implementations available by default: `SimpleXMLTemplateStructureProcessor` – describe a template in XML `SimpleJSONTemplateStructureProcessor` – describe a template in JSON `JSONDummyDataTemplateStructureProcessor` – create dummy data in JSON format that can populate a template. These classes allow application code to perform various operations based on knowing the structure and fields of a given template including creating forms and creating examples of how data structures should look . See the Java API for more information (http://www.docmosis.com/resources)
2	**Template Validation** Templates can now be run through the validation mechanism to detect errors without registering them. The new `DropStoreHelper.validateTemplate()` (also available in `StoreHelper`) method processes a template to determine if there are any errors and returns a `TemplateValidationResult` with error messages and suggested fixes. See the Java API for more information (http://www.docmosis.com/resources)
3	**“Stepped” Repetition** Repeating sections `(<<rs_>>)` and repeating table rows `(<<rr_>>)` directives can now "step" through data in "chunks". This allows 1-dimensional data (eg a List or Array) to be transposed into groups of 2, 3, 4 etc. For example, an array of images can now be laid out in the document in pairs or triplets etc - controlled by the template and no need to change the structure of the data. See the Template Guide for more details (http://www.docmosis.com/resources)

#	Change
4	**Fields can Contain Spaces** Space-padded plain text fields are now identified and processed, eg `<< name >>` (with spaces around "`name`") is still recognized as a field. This helps when using copy and paste of text into fields where word processors may add padding. This can be disabled with the property: `docmosis.analyzer.field.allowPadded=false`
5	**IndexOf() Supported for Strings, Lists and Arrays** The template may use `<<name.indexOf('dave')>>` to get have the applicable `indexOf()` operation invoked for String, List and Array data. For String data, `indexOf('dave')` returns the index of "dave" within the string data (or -1 if not found). For Lists and Arrays, `indexOf()` will return the first index of the element equal to "dave" (or -1 if not found).

API Changes

The following API changes should be noted

Class / Interface	Change
`RenderRequest`	New method to take a template as an `InputStream` - `setTemplateStream()`
`TemplateStructureProcessor`	Added support for creation of data by adding "`index`" parameter to the `repeatBegin()` and `repeatEnd()` methods.

Bug Fixes / Technical Changes

#	Change
1	fixed case where some unmatched section end tags ("<<es_") where not being detected and flagged as an error
2	cleanup of some temporary files being left behind
3	fixed NPE when rendering RTF output on ODT templates with embedded chart objects
4	fixed issue that could cause corrupt document to be produced when trying to write error into document.
5	fixed non-closure of some input streams during processing
6	Fixed possbile exception (NPE) raised when having a rr_ / er_ structure with no rows between.

Docmosis v3.0.6 Release Notes

Dec 2013

New Features

#	Change
1	**Built in DOCX support option (requires Libre Office rather than OpenOffice)** Libre Office 4 DocX support is improving and Docmosis has been upgrade to make some improvements to the DocX result. Docmosis can use Libre Office to produce DocX directly by setting: `docmosis.converter.format.docx.internal.enabled=true` in the `docmosis.properties` file The odf-converter (third party free system) is the alternative for DocX and is still likely to produce better results in general, but requires a separate program to be installed.
2	**Images are now bundled with HTML output** When the rendered document contains images and HTML output is chosen, the images are bundled with the result into a ZIP file. This means that when the HTML result is viewed the images are also displayed. When rendering, if only HTML output is selected, then a HTML text result will be returned (ie no images) by default. To provide the images, a ZIP result is required and this can be requested by: a) Outputting in more than one format (eg HTML and PDF) b) Requesting a ZIP for even a single result (the *compressSingleFormat* flag)
3	**Improved Error Messages for Unsupported Output Formats** If the underlying office engine (Open Office or Libre Office) doesn't support the requested output format, then this is explicitly detected and stated in the error messages. This is important in cases where one engine supports a format and another doesn't (such as with formats XHTML and DocX).

API Changes

The following API changes should be noted

Class / Interface	Change
DataProviderBuilder	JavaDoc updates for example code

Bug Fixes / Technical Changes

#	Change
1	Added UTF-8 decoding of site key from license to allow multi-byte characters in naming
2	Corrected issue where some error message suggestions were displaying XML (eg <text:s text:c="1"/>)
3	updated to leave templates IN the analyzed stream when info about the image can't be determined. Images were being stripped previously.
4	added new (property-disabled) way to treat templates with spanning-rows with "allow row to break" disabled as an error in the template. This will help where such templates are able to crash Open Office and hence fail to render. The property is disabled by default: `docmosis.analyzer.error.nonBreakableRowSpanningRowsFatal=false` since it is rarely an issue.
5	fixed issue where preserved corrupt file was being cleaned up
6	Updated bundled docmosis.properties file to contain examples of settings where Libre Office 4 is in use.

Docmosis v3.0.5 Release Notes

Jul 2013

New Features

#	Change
1	**Automatic processing of DOCX templates by DropStoreHelper** DropStoreHelper previously ignored DocX files when loading templates.
2	**API additions to Reverse Engineer from Templates** New classes *TemplateStructureExtractor* and *TemplateStructureProcessor* provide the ability to interpret the structure of loaded templates and perform arbitrary processing. The *SimpleXMLTemplateStructureProcessor* is an example implementation that dumps the template structure into an XML format.
3	**XML Data Provision Updates** When providing data via XML, text data is loaded in a new way to ensure that data that may have an ambiguous meaning is available under both interpretations. For example: <data> <a>some data</a> </data> Will be loaded into the *DataProvider* such that a field in the template: <<data.a>> will populate with “some data” (this is the new behavior). The following template structure will also populate with “some data”: <<rs_data>> <<rs_a>> <<value>> <<es_a>> <<es_data>> (which is the same as the previous behavior). This simply means XML processing should behave more intiuitively.

API Changes

The following API changes should be noted

Class / Interface	Change
TemplateStoreFactory	The method *getStore(String)* can be passed a custom *TemplateStore* implementation to load. The parameter format is: *custom:<impl class>[:param]* for example: *custom:com.MyStoreImp* or *custom:com.MyStoreImpl:saveLocation1*

Bug Fixes / Technical Changes

#	Change
1	Improved plain text field processing where some fields were not being detected particularly with DOCX format templates.
2	The default example value for the property docmosis.converter.format.docx.external.path in docmosis.properties file has been corrected for linux platforms. It was previously referencing the odf-converter-integrator rather than odf-converter.
3	Javadoc corrected where “wingdings” was being called “windings”.
4	Fixed NPE when <bgcolor> directive in data was used for docmosis fields that were outside of a table in the template.
5	Fixed a problem where Uploading a template cause cause issues with the converter pool if the Upload crashed OpenOffice

Docmosis v3.0.4 Release Notes

Feb 2013

New Features

<table>
<tr><th>#</th><th>Change</th></tr>
<tr><td>1</td><td><b>Improved Literals Processing</b>

Processing of literal values in template fields has been improved. Docmosis can now process numbers, Strings, boolean (true and false) and null values as constants when assigning variables or creating expressions. For example, the following are valid template fields:

<table>
<tr><td>&lt;&lt;cs_{name='Fred'}&gt;&gt;</td><td>test if the value for key "name" is "Fred"</td></tr>
<tr><td>&lt;&lt;cs_{name=null}&gt;&gt;</td><td>test if the value for key "name" is null (undefined)</td></tr>
<tr><td>&lt;&lt;cs_{score<10.5}&gt;&gt;</td><td>test if the value for key "score" is less than 10.5</td></tr>
<tr><td>&lt;&lt;$m=false&gt;&gt;</td><td>set the template-variable "$m" to false</td></tr>
<tr><td>&lt;&lt;cs_{$m=true}&gt;&gt;</td><td>test whether the value for template variable "$m" is true.</td></tr>
</table>
</td></tr>
</table>

API Changes

The following API changes should be noted

None.

Bug Fixes / Technical Changes

#	Change
1	Added default support for OpenOffice installed on Debian Platforms
2	Added default support for LibreOffice 3.6 on linux platforms
3	Lowered default number of document retries to 1 since the Open Office and Libre Office platforms are stable enough that retrying is rarely required
4	Updated field parsing to handle "smart quotes" created by Word 2010 and later
5	Allow template-variables to be set in the cr_ and rr_ rows of tables (where previously they were quietly ignored). This allows variables to be set in rows that are removed during rendering to be defined and used in subsequent processing.
6	Fix - ensure setting the cell colouring via the data only takes effect if the "docmosis.populator.field.markup.process" property is set to true
7	Improved detection of plain text fields where tag contains a quote (') character
8	Fixed issues where only *.docx named files were running through external DOCX converter (and not *.dotx files)

Docmosis v3.0.3 Release Notes

May 2012

New Features

#	Change

1	**HTML-Like Text Markup**
	
	The HTML-Like text markup introduced in 3.0.1 has been extended to allow DATA to control the background colour of table cells. If HTML-like markup is active, then any template-field in a table cell can set the background colour by specifying `<bgcolor="#rrggbb"/>` as the beginning of the data for the field.
	
	For example, given a template field inside a table:
	`<<myName>>`
	If the data contains for key myName contains:
	`<bgcolor="#ff0000"/>James`
	The field will be populated with the text "James" and the table cell colour will be set to red (#ff0000).
	
	The `<bgcolor>` tag must be the first item in the data. There doesn't need to be any textual data to match, so you could have the data and the colour in separate fields. As an example the template might look like this:
	`<<myName>><<cellColour>>`
	And the data could contain have:
	`myName => James`
	`cellColour =><bgcolor="#00ff00"/>`
	and the end result would be a green table cell containing the text James.

API Changes

The following API changes should be noted

None.

Bug Fixes / Technical Changes

#	Change
1	Fixed issue where remote converters could be stalled when something other than Docmosis connects and does the correct handshake.
2	Improved plain text markup processing to capture some cases where Docmosis thought the template was invalid.

Docmosis v3.0.2 Release Notes

April 2012

New Features

None.

API Changes

The following API changes should be noted

Class / Interface	Change
DataProviderBuilder	Improved handing and error reporting for null keys and values.
RenderRequest	Improved javadoc for interaction with ConversionInstruction
ConversionInstruction	Improved javadoc for getConversionFormats() and some PDF-specific methods. Added some new PDF-specific public constants.

Bug Fixes / Technical Changes

#	Change
1	Fixed issue reducing parallel processing using Libre Office. Did not affect Open Office.
2	Improved data processing to correctly compare null and '' (empty string) values between templates and data.
3	Fixed NullPointerException when setting adding null image streams and files and improved messages when null keys used in DataProviderBuilder add() methods.
4	Fixed issue where right-border could disappear when right most columns stripped out of a table using conditional columns and repeating rows. Fixed NullPointerException when a ConversionInstruction passed that has no output formats (ConversionFormat) specified. API documentation improved for ConversionInstruction.
5	Fixed processing to allow RenderRequest settings to work with the contained conversion instruction for determining output formats. API documentation improved for RenderRequest.
6	Updated the shutdown process to provide a grace period for sub-systems to shut down. This fixes spurious errors/warnings during the shutdown process.
7	Updates to some default settings for performance improvements: - io read block size 4k -> 8k - in memory processing limit 8k -> 16k - worker pool max size 10 -> 20 - converter refresh 100 -> 300 - working window for analyser 20 -> 40

Docmosis v3.0.1 Release Notes

March 2012

This release is for most customers a drop-in replacement for the 2.2.2 release. Few parts of the API have incompatible changes, and specific notes are as follows:

1. A new license key is required. License keys for previous versions of Docmosis are not valid. Please visit the Docmosis web site to find out how to obtain your key. If you have purchased a license key within the past 12 months for Docmosis, you will be allowed to upgrade to the new version free of charge.
2. Plain text markup in the templates is turned on by default in the docmosis.properties file that comes with the 3.0.1 release. If you are an existing docmosis user, you will need to add the new properties to your docmosis.properties file if you wish to take advantage of plain text markup. See the new features below for details.
3. HTML-like interpretation of data is disabled by default. This can be enabled in your docmosis.properties file. See the Section below about HTML-like markup.

New Features

#	Change

1	**Plain Text Markup**
	Docmosis fields/placeholders can now be written into the document using plain text rather than Open Office Fields or Word merge Fields. This can make template maintenance much simpler.
	The feature is turned on by default in the downloaded Docmosis bundle. The default delimiters are << and >>. If you are an existing Docmosis user, can add the following properties to your docmosis.properties file:
	`docmosis.analyzer.field.plainText.prefix=<<`
	`docmosis.analyzer.field.plainText.suffix=>>`
	Example1 in the download bundle shows plain text markup in use. It is the same as Example2 except Example1 uses plain text markup and Example2 uses merge fields.
	Please see the latest *Docmosis Template Guide* in the Support section of the Docmosis site for more information.

2	**Image Scaling Options** The template can now indicate three modes of operation when placing images: 1. *stretch* - images are stretched to fit the template placeholder 2. *fit* - images are scaled to fit the template placeholder but maintaining the original aspect ratio 3. *default* - images will be scaled according to the chosen default behaviour which can be overridden on a per-render basis. The use of "`bm_xxx`" to identify an image is now deprecated and replaced by:

#	Change
	1. `img_xxx` - insert the image `xxx` using the default setting (stretch is the pre-defined default). The stretch/fit behaviour can be changed at the system level using the property: `docmosis.analyzer.image.scaling.default=fit|stretch` The behaviour can also be overridden when calling the `DocumentProcessor.render(RenderRequest)` method. 2. `imgstretch_xxx` - insert the image `xxx` stretching the image in x and y directions to fit the template placeholder entirely. 3. `imgfit_xxx` - insert the image `xxx` scaling the image to fit the template placeholder, but preserving the aspect ratio of the image. Please see the latest *Docmosis Template Guide* in the Support section of the Docmosis site for more information.
3	**New PDF and Word Controls** Various new features can be specified (see API changes below) for PDF or Word output including 1. Password Protect 2. Archive Mode and default view for PDF 3. PDF Watermarks Please see the *Docmosis Java API* or the *Docmosis Web Services* Guide in the Support section of the Docmosis site for more information.
4	**XML and JSON Data Support** The engine can now work directly with XML and JSON format data (via the `DataProviderBuilder`) class. Please see the *Docmosis Java API* in the Support section of the Docmosis site for more information.
5	**HTML-like Text Markup** String data can now use a limited set of HTML-style mark-up. This allows bold, italic and underline styles to be applied to a single word, phrase or paragraph. For example, your data can contain: This is <b>bold</b> Which will be rendered as This is **bold** This is disabled by default and is controlled by the property: `docmosis.populator.field.markup.process=true` It can be overridden programmatically using the new `RenderRequest` object for the `DocumentProcessor` API (see API Changes below). See also example1 and example2 in the download which show it in operation for the <<introduction>> placeholder.

#	Change

6	**Experimental DocX Support**
	
	OpenOffice and LibreOffice currently perform poor conversions to and from docx format files. We have enabled Docmosis to interact with the ODF Converter (http://sourceforge.net/projects/oci/) which performs much better conversions.
	
	To use docx support, set the following property in docmosis.properties to indicate where the ODF Converter is installed:
	`docmosis.converter.format.docx.external.path`
	eg:
	`docmosis.converter.format.docx.external.path=c:/program files (x86)/odf-converter-integrator/OdfConverter.exe`
	
	You may find it more convenient to install the ODF Converter Integrator (http://katana.oooninja.com/w/odf-converter-integrator) which provides more cross-platform options. Please note the ODF Converter Integrator may reconfigure your host to open docx files with itself rather than Microsoft Word.

API Changes

The following API changes should be noted

Class / Interface	Change
DocumentProcessor	New render method: `render(RenderRequest request)` allowing all and extended features to be set when rendering. It is intended to be the primary method for use in future. The `RenderRequest` object allows various extra settings including: - using multiple Template Stores - overriding default behaviours including - image scaling behaviour - error handling behaviour - HTML interpretation of text data
DocumentProcessor	`render` methods now return a result object which includes the number of pages and the size of the document produced.
ConversionInstruction	Many new setter methods to allow format-specific properties to be controlled including: PDF password protect PDF watermarking PDF archive mode (PDF/A-1a) PDF default view settings PDF image compression WORD password protect
DataProviderBuilder	New methods to support XML and JSON format data
TemplateIdentifier	A new constructor (String, String) to make it easy to set the name and context at the same time.
ImageScalingDefault	Class defining the constants that can be used for image scaling

TemplateStoreFactory	The method `getStore(String)` has been removed. A single template store location is expected to be configured using system properties. New methods have been added to allow store instances to be obtained with overriding settings for: - whether template errors are fatal (loading into the store results in an exception) or allowed (templates are stored even with errors which can then be shown when rendered) - delimiters to use when analysing templates

Bug Fixes / Technical Changes

#	**Change**
1	Improved processing and reporting of errors in template tables where rows marked as heading rows are used.
2	Improved handling of `TemplateIdentifiers` and `TemplateContexts` to ensure Engine, Cache and `DropStoreHelper` work consistently to identify a template. This corrects a problem where a template update would not be dynamically picked up when rendering.
3	Fixed issue when rendering in multiple formats at the same time - images may not be rendered into final document.
4	Fixed issue where "File In Use" could occur when Update a template on the fly
5	Field Renderers now can be applied in Headers and Footers
6	Fixed NPE that could occur when including an external template and the current template context was "" (empty).
7	Community edition limited to 200 documents per day
8	Improved processing of consecutive spaces which were sometimes condensed to a single space.
9	A `refLookup` that could not find the template name could result in an error that was not meaningful.

[Older] Docmosis v2.2.2 Release Notes

January 2011

New Features

#	Change
1	**New Break Fields** Docmosis now provides 4 fields as a neater way to insert page and column breaks. The first forms will produce a page and column break (for multi-column documents) respectively: *«pageBreak»* *«columnBreak»* The second forms are for use within repeating sections and will insert the corresponding break until the last iteration of the section: *«pageBreakNotLast»* *«columnBreakNotLast»* The new fields can be used instead of or in conjunction with actual page and column breaks in your template. Please see example4 in the download bundle which shows an example use of *«pageBreakNotLast»*.
2	**Image Lookup Extended** Image lookup has been extended to match the facilities of textual data including nesting (eg my.images.image1) and variables ($myImage)
3	**Corrupt Templates are Detected Earlier** Previously if badly corrupted documents were used as templates (for example empty files or incorrect format) then Docmosis would take minutes to identify this. Now such templates are recognised sub-second.
4	**Full-Justification Support for Multi-Line Data** Carriage returns/end-of-line sequences in data now result in the end of a paragraph in the resulting document. This is particularly useful when using full-justified paragraph formats which would previously display overly-spaced lines.

API Changes

The following API changes should be noted

Class / Interface	Change
DataProviderBuilder	New addJavaObject(Object, String, boolean) method has been added to allow "forgiving" mode to be specified during Java reflection for data.
DataProviderBuilder	addJavaObject(Object) has been deprecated since it can mask other data and such behaviour is not obvious.

Bug Fixes / Technical Enhancements

#	**Change**
1	Improved processing and reporting of errors in template tables where rows marked as heading rows are used.
2	Corrected logging of class name where Reflection fails to find any method related to the template field.
3	Fixed issue where images were not substituting in headers under some conditions

[Older] Docmosis v2.2.1 Release Notes

May 2010

New Features

None

API Changes

The following API changes should be noted

Class / Interface	Change
DataProviderBuilder.addJavaObject(Object)	This method has been documented as dangerous due to its ability to hide other data. addJavaObject(Object,String) is the safe alternative.

Bug Fixes / Technical Enhancements

#	Change
1	Reflection now populates correctly from non-List Collections such as Set (eg TreeSet) and Queue.

[Older] Docmosis v2.2.0 Release Notes

April 2010

New Features

#	Change
1	**Template Merging** Docmosis now allows templates to be referenced by other templates and the templates will be merged at processing time. This allows templates to have common content to be separated out into shared templates. For example, if documents have a standard header layout, they can reference a common header template using one of the two new Docmosis fields. This might look like: *«ref:header.doc»* or *«refLookup:headerTemplate»* where the first field would pull in the template called *header.doc* into the current template and the second field would look up the key "*headerTemplate*" in the data provider to get the name of the template to pull in. See the Docmosis Template Guide for more information and *example5* in the download bundle for an example.
2	**New Render Methods** The DocumentProcess class has new overloaded render methods to assist in simplifying document generation. Most notably is the presence of a new Boolean parameter to override the default behaviour of cleaning up the DataProvider after the render. Reusing the data can be helpful if making separate render calls to produce separate documents rather than a single call to produce a zip archive.
3	**Reduced IO** Disk and Network IO has been reduced in the case where the Converters are on the same host as the core engine.
4	**Improved Error and Diagnostic Handling** a) 32/64 bit incompatibilities are detected and suggestions are reported. This is helpful when using a 64 bit Java and there is no 64 bit OpenOffice (such as on Windows and MacOSX). b) Handshaking between the Docmosis core and the converters will report versioning issues, status and environmental differences.
5	**Improved docmosis.properties** The example *docmosis.properties* file has been re-organised to show the critical properties first, better documentation and show some of the other important properties.

API Changes

The following API changes should be noted

Class / Interface	Change
DocumentProcessor	New renderDoc() methods allowing easier access to re-using the same DataProvider between calls. Previously

	only one method allowed this to be specified (within the ConversionInstruction)

Bug Fixes / Technical Enhancements

#	**Change**
1	Repeating Table Rows now allow for conditional rows and hence keep border and background styling features
2	Improvements to rendering errors into the resulting document in corner cases where error was not reported
3	Improvements to ODT (OpenOffice Writer) template processing for end of section detection.
4	Improvements to JavaDoc information

[Older] Docmosis v2.1.1 Release Notes

February 2010

Bug Fixes / Technical Enhancements

#	**Change**
1	OpenOffice 3.2 support OpenOffice has a great new release which has no serious bugs as far as Docmosis is concerned. The previous two production releases (3.0.1 and 3.1.0) had bugs that were not ideal for typical Docmosis use. OpenOffice 3.2 changes a few things under the hood so Docmosis had to upgrade to match.
2	New options to control the way Docmosis loads OpenOffice jars and native libraries. This allows Docmosis to launch the Converters itself rather ("embedded converters") rather than this requiring a separate script (eg runConverter.sh). This means setup is simpler for smaller systems. The launching works from within more Web Application Servers than before, such as JBoss5, Glassfish etc. Note: using "embedded converters" implies you are running everything on the same machine as opposed to a load-distribution configuration. To use embedded converters, update see the information in the example *converterPoolConfig.xml* file. The primary new property allowing control over the library loading is: *docmosis.openoffice.useCustomLoader=true|false* This property defaults to *false* but if you have linkage errors, particularly when using embedded converters, you can try setting this to *true* to load the libraries in a different fashion.
3	The property *template.store.location* can now be left blank, in which case Docmosis will create a temporary area to work with for the template cache.

[Older] Docmosis v2.1.0 Release Notes

November 2009

New Features

#	Change
1	**Hyperlink Insertion** Docmosis can now insert active hyperlinks into documents. A link placeholder is inserted into the template using a name with a "link_" prefix. For example a field *«link_myWebSpace»* will look up the data provider using the key "myWebSpace" and render the result as a hyperlink. Further, the data provided can be delimited using a pipe symbol ("|") to name the link differently from the address. For example, given the field above, a data value of "http://www.mywebspace.com/bluk" would be rendered as a hyperlink and display the text "http://www.mywebspace.com/bluk" in the document. If the data value was "mywebspace|http://www.mywebspace.com/bluk", the link would be rendered into the document displaying the text "mywebspace". See *example1* in the download bundle to see it in action.

API Changes

The following API changes should be noted

Class / Interface	Change
DataProviderBuilder	addFile() methods now allow character encoding to be specified.
StoreHelper	New storeTemplate() method that can take an InputStream as the source of the template rather than just file-based templates.
DropStoreHelper	New process methods to allow Zip and Jar files of templates to be processed directly, or from URLs to Resources from a class loader.

Bug Fixes / Technical Enhancements

#	Change
1	Expressions Enhanced - general improvements i. default boolean true test cs_{isFriend()} was previously invalid, but now evaluates isFriend() as a boolean as expected. ii. null tests supported cs_{getAlpha()=null} iii. size() capability broadened now also applies to any non-reflective data sources iv. isEmpty() capability added cs_{getFriends().isEmpty()}

2	Fixed issue where rs_, cs_ and es_ tags at the first line of a template page could result in extra blank lines in output documents.
3	Added toString() to TemplateAnalysis implementation to allow dumping of analysis information.
4	Fixed issue with nested anonymous er_ tags failing template registration.
5	Fixed issue with second and subsequent Tables Of Content/Tables of Figures etc not being updated correctly.
6	Improved handling of java.sql.Time data type.
7	Subtle fixes to template caching and warnings for large templates
8	Fixes to DataProviderBuilder.addJavaObject(name, Object) so that it works with Collection/Array data types and allows them to be referenced directly by the name given.
9	The connection to OpenOffice will complain with a specific message about unsupported Java version if the OpenOffice API is not compatible with the version of Java in use.
10	Improved population of non-ascii characters to support multiple languages and symbology.