



FirstSpirit™

Unlock Your Content

FirstSpirit™ OfficeConnect FirstSpirit™ Version 5.1

Version	1.7
Status	RELEASED
Date	2015-02-05
Department	Techn. Documentation
Copyright	2015 e-Spirit AG
File name	OFFC_EN_FirstSpirit_OfficeConnect

e-Spirit AG

Stockholmer Allee 24
44269 Dortmund | Germany

T +49 231 . 477 77-0
F +49 231 . 477 77-499

info@e-spirit.com
www.e-spirit.com

e-Spirit

Table of contents

1	Introduction	5
1.1	Overview of the functions.....	5
1.2	Topics covered in this document.....	6
1.3	Procedure for introducing the module.....	7
1.3.1	Use of the rulesets and templates provided.....	7
1.3.2	Creating your own rulesets.....	9
2	Installation	11
2.1	Installing the module on the FirstSpirit server	11
2.2	Installing the project component.....	12
3	Configuring the Module	13
3.1	Checking the license file	13
3.2	Configuring the "OfficeImportService" service	15
3.3	Configuring the "OfficeImportProject Configuration" project component.....	16
4	XML Definition of the Rulesets	18
4.1	XML syntax.....	21
4.2	Structure, nesting and inheritance	21
4.3	Tags.....	25
4.3.1	ImportRuleSets.....	25
4.3.2	mapping.....	25
4.3.3	style.....	28
4.3.4	element.....	30



4.3.5 attribute.....	31
4.3.6 text.....	33
4.4 Attributes.....	35
4.4.1 name.....	35
4.4.2 mapname.....	35
4.4.3 tag.....	36
4.4.4 maptag.....	36
4.4.5 handler.....	37
4.4.6 default.....	39
4.4.7 id.....	40
4.4.8 findtag.....	41
4.4.9 content.....	41
4.4.10 mediaref.....	41
4.4.11 inherit.....	41
4.4.12 break.....	44
4.4.13 mapattributes.....	45
4.4.14 value.....	46
4.4.15 mapvalues.....	48
4.4.16 class.....	49
4.5 Handling unknown tags.....	50
4.6 Example: Creating a rule to reproduce an individual Word format style sheet	50
4.6.1 Step 1: Preparing the Word document.....	50
4.6.2 Step 2: Creating the necessary format template in FirstSpirit.....	50
4.6.3 Step 3: Outputting the Word document in HTML.....	52
4.6.4 Step 4: Analysis of the HTML.....	52
4.6.5 Step 5: Creating the XML rule definition.....	52
4.7 Default ruleset of the FirstSpirit OfficeConnect module.....	53



5	Configuring the Input Components	57
5.1	Activating the import function.....	57
5.2	Limiting selectable rulesets	58
5.3	Restrictions (format and link templates, lists, tables)	59
5.3.1	Format templates.....	59
5.3.2	Link templates.....	59
5.3.3	Lists.....	60
5.3.4	Tables (only DOM-Editor)	60
5.4	Using link templates	61
5.5	Output of lists.....	63
6	Editorial Work in SiteArchitect.....	66
6.1	Explanations concerning the Word document.....	66
6.2	Importing into FirstSpirit SiteArchitect.....	68
6.3	Importing formatted texts	70
6.3.1	Text from Word documents	70
6.3.2	Text in the DOM Editor	71
6.3.3	Exemplary HTML output.....	73
6.4	Importing lists.....	73
6.4.1	Lists from Word documents.....	74
6.4.2	Lists in the DOM Editor.....	75
6.5	Importing links	76
6.5.1	Links from Word documents	76
6.5.2	Links in the DOM Editor.....	77
6.6	Importing pictures.....	79
6.6.1	Inserting into the DOM editor.....	79
6.6.2	Creating pictures in the Media Store	80



6.7	Importing tables	81
6.7.1	Tables from Word documents.....	81
6.7.2	Tables in the DOM Editor.....	83
7	Legal notices	84



1 Introduction

The FirstSpirit™ OfficeConnect module is an importer for the Microsoft Office HTML clipboard format used by Microsoft Word, Excel, and PowerPoint. The module translates information from the Microsoft Office document into information that FirstSpirit is able to process and display. Specifically, meta information from the Word document (e.g., formatting) is linked with formatting and link templates in FirstSpirit. This is possible as the standard document exchange format for MS Office applications is HTML. The module accesses individually definable sets of rules that define which HTML information from the Word document is assigned to which format or link template in FirstSpirit.

This ultimately allows text, formats, section changes, images, links, and tables to be transferred from Microsoft Office documents into FirstSpirit SiteArchitect, where they can be edited further. The following documentation uses the transfer of content from Microsoft Word documents as an example of this.



*The "FirstSpirit™ OfficeConnect" functionality is **not** supported in ContentCreator.*



This document is provided for information purposes only. e-Spirit may change the contents hereof without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. e-Spirit specifically disclaims any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. The technologies, functionality, services, and processes described herein are subject to change without notice.

1.1 Overview of the functions

The OfficeConnect function can be used in the

- DOM Editor (CMS_INPUT_DOM) and
- DOM table (CMS_INPUT_DOMTABLE)

input components.





Unless specified otherwise, the whole of this document mainly relates to both input components. Alternatively, they are also jointly named "DOM input components" in the following.

The following information from Word documents is taken into account on importing into FirstSpirit:

- Continuous text
- Character formatting (bold, italics, underlined and combinations thereof)
- Paragraph formatting (e.g. headings)
- Paragraph breaks and "soft" line breaks (<RETURN> or <SHIFT> + <RETURN>)
- Lists (with numbers or symbols (bullets), for restrictions on using lists and Word 2010, please see Chapter 6.4 page 73)
- Pictures (including automatic upload into Media Store)
- Links
- Tables (including merging, cell colour, cell alignment and cell format)



To ensure the Office module works correctly, continuous text from Word documents should only be inserted into the DOM editor or into inline tables, and tables from Word should only be inserted into the DOM table or into inline tables. Otherwise, inserted content can no longer be edited.

1.2 Topics covered in this document

Chapter 2: Describes installation of the "FirstSpirit OfficeConnect" module on the server and installation of the project component in a project (from page 11).

Chapter 3: Explains the server and project-wide configuration options provided by the module (from Page 13).

Chapter 4: Uses examples to describe the syntax used for XML definition of the rulesets, with which content from Word documents can be reproduced in the DOM Editor (CMS_INPUT_DOM input component) or in the DOM Table (CMS_INPUT_DOMTABLE input component) (from Page 18).



Chapter 5: The DOM Editor and / or the DOM Table must be adjusted by the template developer so that the module can be used in SiteArchitect (from Page 57).

Chapter 6: This chapter is primarily intended for FirstSpirit editors and illustrates how the module can be used in SiteArchitect for editorial purposes. To this end, by way of example, the Word test document supplied is imported step by step and the effect in the DOM Editor and DOM Table as well as in the HTML output is shown. This chapter is also interesting for development of the XML rulesets (from Page 66).

1.3 Procedure for introducing the module

1.3.1 Use of the rulesets and templates provided

The default configuration of the OfficeConnect module contains two rulesets which can be used immediately:

- "Project local Import Ruleset (generic link)": ruleset for converting content from Word documents with "generic" links
- "Project local Import Ruleset": ruleset for converting content from Word documents with "static" links (for projects working with static links (FirstSpirit versions < 5.0))
- "Standard (text only import)": ruleset for content which is not to be converted



If HTML content (e.g., from a browser or from the integrated preview) is to be used in a DOM editor with an Office module function via drag-and-drop, there must be a corresponding set of rules with the name `Default`.

The rules defined in the global set of rules make it possible to import the Word test document that is also available with all formatting, links, and images into the DOM editor or the DOM table.






On the one hand, formatting from Word is displayed using the format templates available in the standard FirstSpirit scope of delivery (in the Template Store in SiteArchitect), e.g., **bold** (format template tag "b"), *italic* (format template tag "i"), or underlined (format template tag "u").

On the other hand, the following format templates should also be included in the project with the following properties:



- **H1-5:** Section format templates for headings of levels one to five:
 - Tag: "h1" to "h5"
 - Style: "Bold"
 - Sizes: "20", "18", "16", "15", "13"
 - HTML channel: `<h1>${CMS_VALUE(#content)}</h1>`
- **Note:** Section format template for information in red font:
 - Tag: "note"
 - Color: "#FF0000"
 - HTML channel: `${CMS_VALUE(#content)}`
- **Important:** Character format templates for information in red font:
 - Tag: "important"
 - Color: "#FF0000"
 - HTML channel: `${CMS_VALUE(#content)}`
- **s:** Character format template for strikethroughs:
 - Tag: "s"
 - Style: "Strikethrough"
 - HTML channel: `<s>${CMS_VALUE(#content)}</s>`

Furthermore, the module uses the FirstSpirit default format templates "List" and "List entry" for lists and bulleted lists (tags "ul" and "li") and the default format templates "Table", "Table row", and "Table cell" for tables (tags "table", "tr", and "td"). For restrictions on using lists and Word 2010, please see Chapter 6.4 page 73

Note on using the Office module with inline tables: The Office module can also be used in inline tables (DOM editor with parameter `table="yes"`, for more information see *FirstSpirit Online Documentation*, "Templates (Basics)" / "Structure of templates" / "Inline table templates"). The cells in a table imported from Word can be edited using the functions available in the relevant DOM editor. Cells can, however, only be edited using the  icon if the formatting in the Word document (background color, font color, alignment) corresponds to the display rules of the table format template that form the basis of the inline table (see Chapter 4.4.14, page 46). The minimum and maximum number of rows and columns defined in the table format template are not taken into consideration for the import. This means that if the maximum number of rows is 6, a table from Word with 10 rows is imported with all 10 rows, and all cells can be edited as usual. However, additional rows cannot be added. If the table is reduced to 6 rows, additional rows cannot be added. Rows that were previously deleted within a session can however be restored using the   or   icons.

Furthermore, special **link templates** are required to import links and images. The templates required for importing the Word test document are located in the "Mithras Energy" demo project



under the "Link templates" node in the Template Store:

- **textlinkinternal**: Link template for displaying imported images
- **textlinkexternal**: Link template for displaying external links

In this case, when importing into the DOM editor (see Chapter 6.2, page 68 and Figure 6-2), the "Project local Import Ruleset (generic link)" default set of rules must be selected, otherwise the following message is displayed: "The link template of the import module defined in the server/project configuration could not be loaded. (textlinkinternal)".



For the links to be displayed correctly in FirstSpirit, you must ensure that special variable identifiers are used (see Chapter 5.4, page 61).



The appearance of the format and link templates used in SiteArchitect can be adapted to the requirements of the relevant project for the editor and when it is published on the website. For more information on the configuration options, see FirstSpirit Online Documentation, section "Templates (Basics)"/"Structure of templates"/"Inline table templates".

For more information on the Word document for the test import, see Chapter 6.1, page 66.

1.3.2 Creating your own rulesets

Use of the following procedure is recommended to define your own rules or rulesets:

1. First, it is necessary to analyse the word documents to be imported and to create a list of the section (paragraph) and character formats used.
2. In a second step, it is necessary to check whether the section and character templates (paragraph and character style sheets) are used consistently in these Word documents. Depending on the individual configuration of Word and personal way of working, different style sheets (format templates) can be contained in a Word document, although the layout and function of these templates resemble each other strongly. To achieve the best possible result for the Word import, the style sheets in Word should be consolidated first, because the fewer the number of different style sheets there are in a document, the easier it is to create the rulesets for the subsequent import.



3. A corresponding FirstSpirit section and character format template (area "Format templates" in the Template Store of the SiteArchitect) should be available for each paragraph and character format contained in the Word document. If they are not yet available, these templates should be created in FirstSpirit (see FirstSpirit Online documentation, "Templates (basics)" / "Composition of templates", Chapter: "Format Templates"). The "Properties" tab is used to define how texts with this formatting are displayed in the DOM Editor; the "HTML" tab (or tabs for other output channels) is used to define how texts with this formatting are output on the website (or in other output media). If applicable, it is advisable to create a table as shown in Chapter 6.1 page 66.

In addition, an internal link template must be available for importing pictures; this template contains an input component for selecting a media object (e.g. `FS_REFERENCE`). An external link template must be available for importing links.

4. The HTML information which has to be translated for importing into FirstSpirit can be determined by exporting the Word document in HTML format. The XML rule definitions must then be created on the basis of this information (see Chapter 4 page 18, in particular Chapter 4.6 page 50).
5. The FirstSpirit input components to be used to import Word documents (DOM Editor, DOM Table) must then be configured for the OfficeConnect function. To do this, among other things, the format templates created in the 4th step must be integrated in the input components in which the OfficeConnect function is to be available (see Chapter 5.3 page 59).

The actual data import can then take place (see Chapter 6 page 66).



The configuration of a set of rules only ever affects the current import. Subsequent changes to a set of rules do not affect previously imported content.



2 Installation

2.1 Installing the module on the FirstSpirit server

The "FirstSpirit OfficeConnect" module is a licensable function. First, it must be installed in the FirstSpirit ServerManager. To do this, the "Modules" menu entry is selected in Server Properties. Click the "Install" button to open a file selection dialog. The fsm file to be installed (file name "fs-office.fsm") can be selected here. The successfully installed module file is then displayed in the list of installed FirstSpirit modules with the name "FirstSpirit Office":

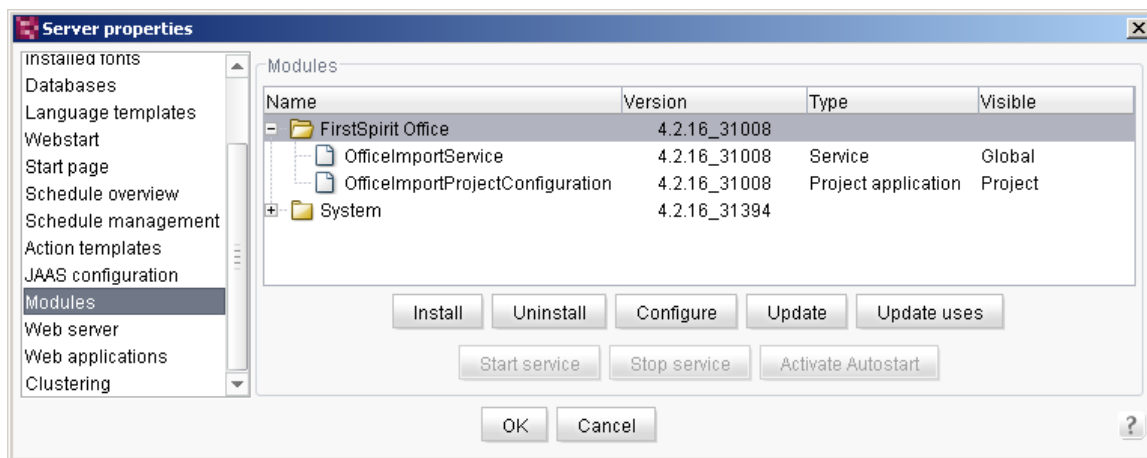


Figure 2-1: Installation of the "FirstSpirit OfficeConnect" module

The module's components are the "OfficeImportService" service and the "OfficeImportProjectConfiguration" project application. For further information on this dialog, see *FirstSpirit Manual for Administrators*.

The "OfficeImportService" **service** enables the central definition of rules by which Word documents are imported. It is available globally, i.e. server-wide, in each project located on the FirstSpirit server. After starting the service, the OfficeConnect function for the whole server can be configured (see Chapter 3.1 page 13).

The "OfficeImportProjectConfiguration" **project application** is a "local project" component. Following installation, it can be added to the required projects via their project properties (see Chapter 2.2 page 12) and the OfficeConnect function can be configured for each specific project (see Chapter 3.3 page 16).



2.2 Installing the project component

If other or extending rules are to be used in a project, the "OfficeImportProjectConfiguration" project component must be installed in the required project. To do this, the "Project Components" menu entry within the project properties is opened.

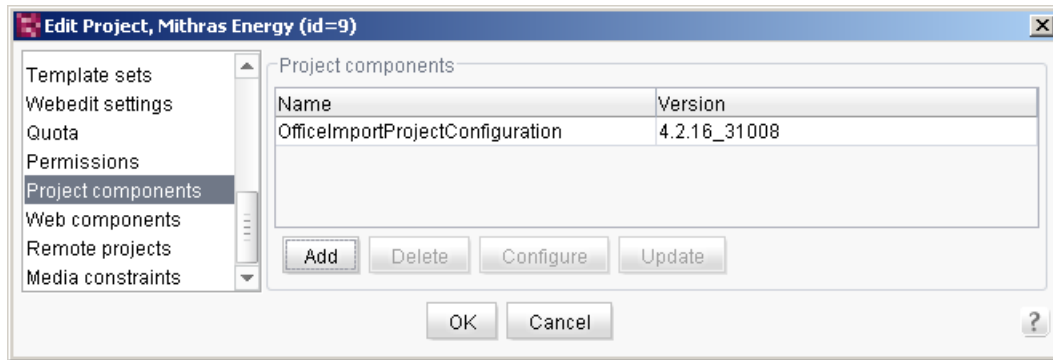


Figure 2-2: Installing the project component

Add: Click the button to open the "Add" dialog. The list shows all project components installed on the server (see Chapter 2.1 page 11). Select the "OfficeImportProjectConfiguration" entry. The globally configured functions of the installed component are then available in the project and can be further configured (see Chapter 3.3 page 16).

If the project component is not added, the global rules only are available in the SiteArchitect.

For further information on this dialog, see *FirstSpirit Manual for Administrators*.



3 Configuring the Module

The following three options are available for configuring the module:

- To globally design one or several rulesets for **all** the FirstSpirit projects of a server, it is only necessary to configure the "OfficeImportService" accordingly (see Chapter 3.2 page 15).
- The "OfficeImportProject Configuration" project component must be configured accordingly, in order to define one or several individual rulesets for **one** FirstSpirit project – independent of the global rulesets applicable server-wide. The "Extend the global ruleset configuration" option must be disabled (see Chapter 3.3 page 16).
- The "OfficeImportProject Configuration" project component must be configured accordingly, in order to use the global rulesets defined server-wide for **one** FirstSpirit project and to additionally extend it to include individual rulesets. For this, the "Extend the global ruleset configuration" option must be enabled (see Chapter 3.3 page 16).



The selection of sets of rules that the editor can use in a project can also be restricted by the input component configuration (see Chapter 5.2, page 58).

3.1 Checking the license file

The applicable FirstSpirit functions of the license file `fs-license.conf` are shown via the FirstSpirit Server Monitoring menu "FirstSpirit – Configuration – License". The parameter `license.OFFICE_IMPORT` has to be set to a value of `1` to use the Office functionality (see Figure 3-1).

A valid license can be requested from the manufacturer and is added in the blue section of the window. The **Save** button is used to save the license file.



Tampering with `fs_license.conf` will result in an invalid license. If changes become necessary, please contact the manufacturer.

The server does not have to be restarted when inserting a new `fs_license.conf` configuration file. The file is updated on the server automatically.




License

```
license.ID=2226
#FIRSTspirit license
#Tue Jun 24 10:11:49 CEST 2014
license.USER=e-spirit
license.EXPDATE=15.01.2015
license.MAXPROJECTS=0
license.MAXSESSIONS=0
license.MAXUSER=0
license.SOCKET_PORT=0
license.VERSION=5
license.MODULES=personalisation,search,integration,newsletter,portal,form_edit,enterprise_search
license.WEBEDIT=1
license.WORKFLOW=1
license.REMOTEPROJECT=1
license.PACKAGEPOOL=1
license.DOCUMENTGROUP=1
license.ACCESS_API=1
license.APPTAB_SLOTS=20
license.ARCHIVE=0
license.CLUSTERING=1
license.ENTERPRISE_BACKUP=1
license.HIGHAVAILIBILITY=1
license.OFFICE_IMPORT=1
license.OFFICE_INTEGRATION=1
license.SCOPE=CORPORATE
license.TYPE=PRODUCTION
```

Figure 3-1: Parameter display in the license file (server monitoring)



3.2 Configuring the "OfficeImportService" service

Click the "OfficeImportService" service and the  button to open the service's configuration dialog:

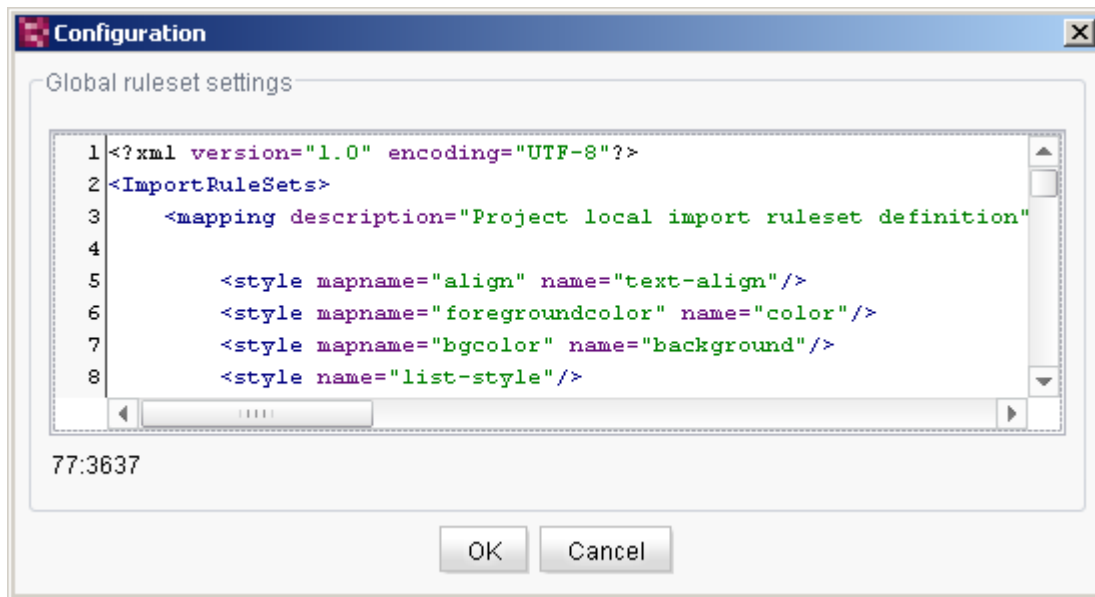


Figure 3-2: Service configuration



The service must be started first, before it can be configured. If it is not, the "Start service?" message appears as a prompt. Click "Yes" to start the service and the configuration dialog appears (see Figure 3-2).



The service can also alternatively be configured and controlled using FirstSpirit Server Monitoring, via "FirstSpirit" / "Configuration" / "Services" or "FirstSpirit" / "Control" / "Services".

This text field can be used to enter global rules for the import of texts from Word documents, which apply to the whole server. These rules define how the formatting from the Word document subsequently has to be converted into FirstSpirit expressions. For detailed information on XML definition of the rulesets, see Chapter 4 page 18.




Different rulesets can be defined for different purposes or different document types. For example, one ruleset can be defined for technical documents and one for marketing documents. The name (`name` attribute) and description (`description` attribute) of the rulesets are subsequently displayed to the editor while working with the module in SiteArchitect and are offered for selection (see Figure 6-2).

3.3 Configuring the "OfficelmportProject Configuration" project component



The service must be started first before the "OfficelmportProject Configuration" project component can be configured (see Chapter 2.1 page 11).

Click the "OfficelmportProjectConfiguration" project component and the  button to open the project component's configuration dialog:

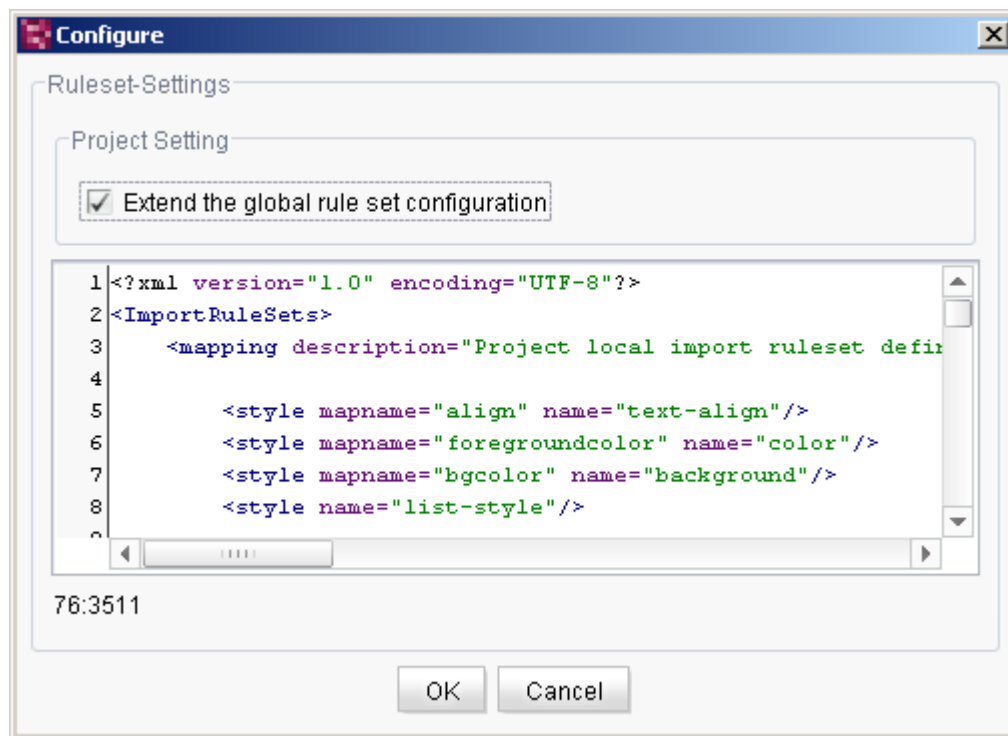


Figure 3-3: Configuring the project component

This dialog can be used to configure special import rules for the respective project. Here too, the text field can be used to define various rulesets for different purposes (see Chapter 4 page 18).

Extend the global ruleset configuration: This option can be used to define whether the server-wide configured rules (see Chapter 3.1 page 13) are to be replaced by the rules specified here or



whether project-specific rulesets are to be added. If the option is enabled, in addition to the rules defined for server-wide application, the project-specific rules defined here are also subsequently offered to the editor for selection while they are working with the module in SiteArchitect. If the option is disabled, the server-wide defined rules are not available to the editor; they can only use the project-specific rules.



If the import function is not available across the whole server, but only for (one) specific project(s), the text field of the "OfficeImportService" service (see Figure 3-2) is left empty and the required set of rules is only entered in the text field of the project component (see Figure 3-3). The selection of sets of rules that the editor can use in a project can also be restricted by the input component configuration (see Chapter 5.2, page 58).



4 XML Definition of the Rulesets

If content from Word documents is copied to the clipboard using <CTRL> + <C>, it is mainly stored in pure HTML format. In addition, the clipboard also contains extensive Word-specific information, which cannot be used for the import into FirstSpirit. This HTML information can, e.g. be displayed by saving the Word document concerned as an html file and then opening it in a browser using the "Show Source Text" option, for example:

```
<html>

<head>

<meta http-equiv=Content-Type content="text/html; charset=windows-1252">
<meta name=Generator content="Microsoft Word 11 (filtered)">

<style>

<!--

/* Font Definitions */

@font-face

{font-family:Wingdings;
panose-1:5 0 0 0 0 0 0 0 0 0;}

/* Style Definitions */

p.MsoNormal, li.MsoNormal, div.MsoNormal

{margin-top:0cm;
margin-right:0cm;
margin-bottom:0cm;
margin-left:99.0pt;
margin-bottom:.0001pt;
text-indent:-18.0pt;
font-size:10.0pt;
```



```
font-family:Arial;}

...

/* List Definitions */

ol

{margin-bottom:0cm;}

ul

{margin-bottom:0cm;}

-->

</style>

</head>

<body lang=DE>

<div class=Section1>

<p class=MsoNormal>&nbsp;</p>

</div>

</body>

</html>
```

The rulesets used in FirstSpirit OfficeConnect are there for linking the HTML information from the clipboard (e.g. <p> for paragraphs, for bold, <h1> for level one headings, <td> for table cells, for pictures, for links, etc.) with the format templates (for character and section formatting, lists, tables, inline tables etc.) and the link templates (for pictures and links) in FirstSpirit. As a result, for example, "bold" character formatting from Word is "translated" into the



"bold" character formatting of the FirstSpirit DOM Editor. The result of this is, on the one hand, that bold text from a Word document is also formatted and displayed as bold text in the DOM Editor, as defined on the "Properties" tab of the relevant format template (Template Store). The FirstSpirit format template also defines how the "bold" character formatting is to be output (e.g. in the HTML channel, "HTML" tab of the format template concerned).

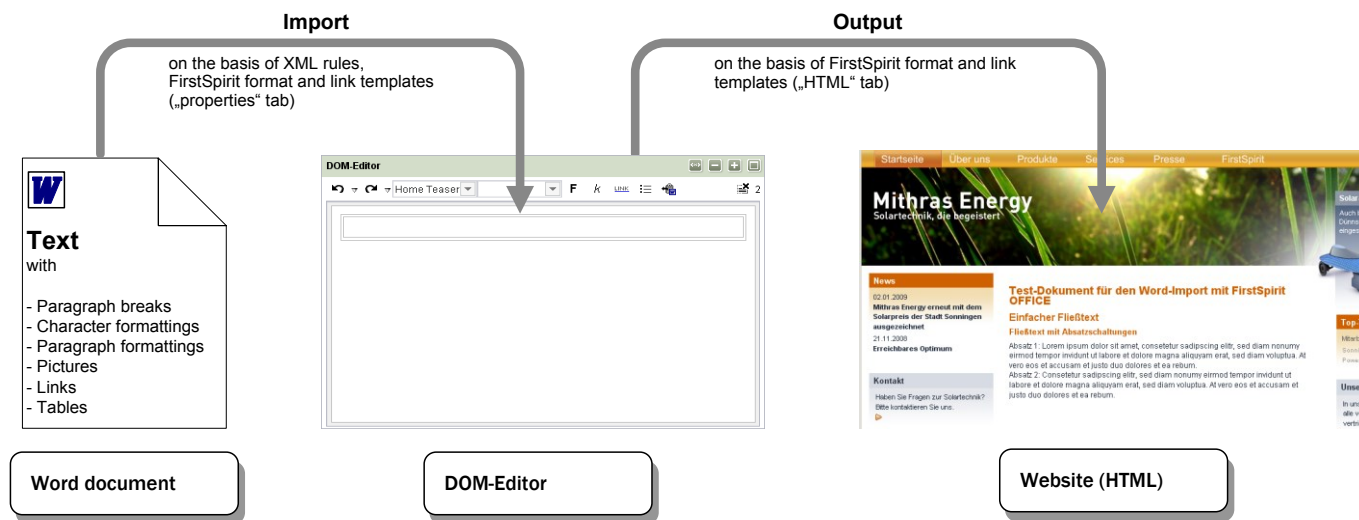


Figure 4-1: Schema import – Output

The rulesets in FirstSpirit OfficeConnect are configured in XML format.



4.1 XML syntax

XML documents must be well-formed and valid, so that they can be read out and interpreted by a parser. The following rules must be followed:

- **XML declaration:** An XML declaration must be specified at the start of the XML definition. Example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

- **Structure:** For valid, well-formed XML, all opened tags must also be closed again.
- **Comments:** Comments can be defined with `<!-- . . . -->`. Everything contained within the angle brackets is ignored in the import.



When non well-formed XML is saved, a corresponding error dialog is displayed and the document cannot be saved.

4.2 Structure, nesting and inheritance

The basic structure of the XML rule definition consists of

- the XML declaration (see Chapter 4.1 page 21) and
- the root element (see Chapter 4.3.1 page 25).

It is followed by a `mapping` element for each ruleset (see Chapter 4.3.2 page 25). Within this element, an XML tag must be specified for each piece of information with special formatting originating from the Word document (see Chapter 4.3.3 to 4.3.6 from page 28). This can be:

- `style` (see Chapter 4.3.3 page 28)
- `element` (see Chapter 4.3.4 page 30)
- `attribute` (see Chapter 4.3.5 page 31)
- `text` (see Chapter 4.3.6 page 33)



All elements (apart from the root element) must be assigned attributes, namely the following:

for the `mapping` tag: (see Chapter 4.3.2 page 25).

- `description`
- `linkConfigExternal`
- `linkConfigInternal`
- `mimeType`
- `name`
- `versionTag`

for the `style` tag: (see Chapter 4.3.3 page 28).

- `mapname`
- `name`

for the `element` tag:

- `name` (see Chapter 4.4.1 page 35)
- `mapname` (see Chapter 4.4.2 page 36)
- `tag` (see Chapter 4.4.3 page 36)
- `maptag` (see Chapter 4.4.4 page 36)
- `handler` (see Chapter 4.4.5 page 37)
- `default` (see Chapter 4.4.6 page 39)
- `id` (see Chapter 4.4.7 page 40)
- `findtag` (see Chapter 4.4.8 page 41)
- `content` (see Chapter 4.4.9 page 41)
- `mediaref` (see Chapter 4.4.10 page 41)
- `inherit` (see Chapter 4.4.11 page 41)
- `break` (see Chapter 4.4.12 page 44)
- `mapattributes` (see Chapter 4.4.13 page 45)
- `value` (see Chapter 4.4.14 page 46)
- `mapvalues` (see Chapter 4.4.15 page 48)
- `class` (see Chapter 4.4.16 page 49)



for the `attribute` tag:

- `mapname` (see Chapter 4.3.5 page 31)
- `name` (see Chapter 4.3.5 page 31)
- `handler` (see Chapter 4.4.5 page 37)

for the `text` tag: (see Chapter 4.3.6 page 33)

- `action`

The examples given in the following are based on the rulesets supplied in the module's default configuration.

The fundamental assignment of HTML information from the Word document to format and link templates in FirstSpirit is defined by `element` tags (see Chapter 4.3.4 page 30). The `element` tags in turn contain tags (`attribute`, `text` or other `element` tags), which define the conversion rules for sub-elements. They can be nested to any depth (lower level) required. However, for improved clarity, the nesting depth should be kept small and the XML code should be kept as "lean" as possible.

Sub `element` tags inherit the attributes of the higher-level `element` tags as well as of the `attribute` and `text` tags. This inheritance can be inhibited by the `inherit="NONE"` attribute (see Chapter 4.4.11 page 41). An `element` tag can also inherit the attributes of another `element` tag which is not on a higher-level. To do this, a name must be given to the `element` tag from which the attributes are to be adopted using `id="..."`. The `element` tag which is to inherit the attributes can access the attributes under this name via `inherit="ID_OF_THE_INHERITING_ELEMENT"`. This is relevant, e.g. for table cells, whose contents are to be converted in the same way as continuous text content.



Example:

```
<element default="true" handler="map" id="HTML.paragraph" tag="p">
...
<element handler="table" tag="table">
  <element tag="tr">
    <attribute name="colspan"/>
    <attribute name="rowspan"/>
    <attribute handler="style" name="style"/>

    <element inherit="HTML.paragraph" tag="td">

      <element handler="strip" inherit="HTML.paragraph"
        mapattributes="true" tag="p"/>
    </element>

  </element>

</element>

</element>
```

In this example, the attributes of the first line of the HTML tag `p` of a continuous text (body) paragraph are also used for table cells (HTML tag `td`) through the `id` "HTML.paragraph" and paragraphs within a table cell (HTML tag `p` within the table, which is opened with `tag="table"`).

The rough structure of the XML ruleset is based on the structure of the HTML file from the Word clipboard and therefore includes, among other things, the following elements:



- Head
- Body
- Paragraphs (<p>)
 - Pictures
 - Breaks (
)
 - Character and paragraph formatting
 - Links
 - Lists
 - Tables
 - Table rows
 - Table cells
 - Headers (<th>)

This is also the schema of the ruleset example supplied with the module. The indenting below the rules for paragraphs shows that all indented elements (pictures, tables, etc.) inherit the rules for paragraphs.

4.3 Tags

4.3.1 ImportRuleSets

The XML definition must be enclosed by the root element

```
<ImportRuleSets>... </ImportRuleSets>
```

The `ImportRuleSet` tag does not require any attributes.

4.3.2 mapping

The root element can contain several rulesets. Each ruleset is opened with a mapping element

```
<mapping...>
```



Various attributes can be specified for each mapping element:

Attribute	Explanation / Example	Mandatory attribute
description	<p>Specification of a description for the ruleset; this is displayed to the editor in the DOM input component when they work with the OfficeConnect module (see Figure 6-2), e.g.</p> <pre>description="Project local import ruleset definition"</pre>	yes
linkConfigExternal	<p>Specification of the unique name of the link template that should be used when importing Word documents for displaying external links, e.g.,</p> <pre>linkConfigExternal="textlinkexternal"</pre> <p>This link template must also be specified if the use of certain link templates is defined in the relevant DOM input component through the <i>LINKEDITORS / LINKEDITOR</i> tags. This attribute does not need to be specified if no links are to be converted when importing from Word documents.</p>	no
linkConfigInternal	<p>Specification of the unique name of the link template, which is to be used for the import of media in Word documents, e.g.</p> <pre>linkConfigInternal="textlinkinternal"</pre> <p>This link template must also be specified if the use of certain link templates is defined in the relevant DOM input component through the <i>LINKEDITORS / LINKEDITOR</i> tags. This attribute does not have to be specified if media is not to be imported.</p>	no



Attribute	Explanation / Example	Mandatory attribute
mimeType	<p>Specification of the MIME type, e.g.</p> <pre>mimeType="text/html"</pre> <p>The "Standard (text only import)" ruleset supplied with the module is intended to be used to import texts, which are not to be converted when imported in the installed OfficeConnect module. It therefore has the MIME type <code>mimeType="text/plain"</code>.</p>	yes
name	<p>Specification of a unique name for the ruleset; this is displayed to the editor in the DOM Editor when they work with the OfficeConnect module (see Figure 6-2), e.g.</p> <pre>name="Project local Import Ruleset"</pre> <p>If HTML content (e.g., from a browser or from the integrated preview) is to be used in a DOM editor with an Office module function via drag-and-drop, there must be a corresponding set of rules with the name <code>Default</code>.</p>	yes
versionTag	<p>This attribute specifies the version number of the XML ruleset. It is automatically increased by one each time a change is made, e.g.</p> <pre>versionTag="24"</pre> <p>With each import, this version number is used to initially check whether changes have been made to the rulesets since the last import. If not, the rulesets still in the client cache are used directly. If changes have been made in the meantime, the modified rulesets are loaded first and are then applied.</p>	automatic



The mapping element must be closed again at the end of the XML definition.

Example of several rulesets:

```
<?xml version="1.0" encoding="UTF-8"?>

<ImportRuleSets>

  <mapping description="Project specific ruleset"
    linkConfigExternal="textlinkexternal"
    linkConfigInternal="textlinkinternal"
    mimeType="text/html" name="Project" versionTag="24">

    ...

  </mapping>

  <mapping description="Project specific ruleset for plain text"
    linkConfigExternal="textlinkexternal "
    linkConfigInternal="textlinkinternal "
    mimeType="text/html" name="Text" versionTag="24">

    ...

  </mapping>

</ImportRuleSets>
```

After the root and mapping element, an assignment rule by which the information is to be reproduced in FirstSpirit is defined for each piece of information copied onto the clipboard from a Word document.

4.3.3 style

The `style` tag is used to define the conversion of "style" tags from the HTML of the Word document. The conversion rules defined with the help of the `style` tag apply globally to the whole Word document.

The conversion of "style" tags from the Word document into information which can be evaluated in FirstSpirit is established within a `style` tag by the `mapname` and `name` attributes: `name` gives the values of the "style" tags from the Word document which are to be converted when imported into FirstSpirit; `mapname` is used to specify the FirstSpirit value into which the HTML attribute is to be translated. Both attributes must exist in a `style` tag, e.g.



```
<style mapname="align" name="text-align"/>
```

In this example, information from the Word document, which is marked up by the "style" tag `text-align`, is linked with the FirstSpirit `align` attribute on being imported into a DOM input component. With this, on being imported into a DOM input component, text alignments in Word, e.g. `right` (flush-right/right-aligned) or `center` (centered) are reproduced in `align`, which is used to control text alignment in FirstSpirit.

A `style` tag must be defined in the XML ruleset for each "style" tag from the Word document to be evaluated in FirstSpirit.

If `mapname` is not explicitly specified, the name given by the `name` attribute is used. Example:

```
<style name="list-style"/>
```

In this example, the "list-style" value defined for `name` is used as the `mapname`. This is therefore an alternative, shortened notation for

```
<style mapname="list-style" name="list-style"/>
```

Attribute	Explanation / Example	Mandatory attribute
name	Specification of "style" tags from the Word clipboard, e.g. <code>name="text-align"</code> (see also Chapter 4.4.1 page 35)	yes
mapname	Specification of a value in FirstSpirit, into which the <code>name</code> attribute is to be translated, e.g. <code>mapname="align"</code> If the values of the "style" tag from the Word document and the corresponding value in FirstSpirit are identical, it is not necessary to specify <code>mapname</code> . (see also Chapter 4.4.2 page 35)	no



No other attributes have to be specified for the `style` tag.

4.3.4 element

The `element` tag is used to define the conversion rules for the HTML elements of the Word document, e.g. elements for

- the HTML framework: `html`, `head`, `body`
- Text formatting: `p`, `div`, `span`, `br`, `b`, `strong`, `i`, `h1`, `li`, `ol`, `ul`
- the definition of links: `a`
- the definition of tables: `table`
- the integration of graphic pictures: `img`

An `element` tag **must** be specified in the XML definition for each HTML element contained in the Word document. If no `element` tag is specified, the so-called default element is used (see Chapter 4.4.6 page 39).

The following **mandatory attributes** must be specified for the `element` tag:

- `tag` / `maptag`: These attributes are used to establish the link between the HTML information from the Word document and the FirstSpirit format and link templates: `tag` is used to specify the values of the "style" tags from the Word document which are to be converted on being imported into FirstSpirit; `maptag` specifies the FirstSpirit value (abbreviation of the FirstSpirit format template or other values which are used internally to control FirstSpirit), into which the HTML element is to be translated (see also Chapter 4.4.3 page 36 and Chapter 4.4.4 page 36). `maptag` does not need to be specified if the same name is used as that given for the `tag` attribute.
- `handler`: This attribute defines how the HTML information from the Word document is to be processed (handled). It is also required to import pictures, links and tables (see Chapter 4.4.5 page 37).

The following **optional attributes** can be used with `element`:

- `default` (also called "default element") Definition of conversion rules for unknown information from Word documents (see Chapter 4.4.6 page 39)
- `id` Definition of names for `element` tags, in order to be able to reference to them elsewhere (see Chapter 4.4.7 page 40)



- `findtag` Read out tags from the Word document (in conjunction with `handler="find"`), e.g. the title of the Word document (see Chapter 4.4.8 page 41)
- `content` Process / Parse objects from Word documents, e.g. pictures (in conjunction with `handler="media"`) (see Chapter 4.4.9 page 41)
- `mediaref` Import pictures (in conjunction with `handler="media"`) (see Chapter 4.4.10 page 41)
- `inherit` Control the inheritance of attributes (see Chapter 4.4.11 page 41)
- `break` Convert text breaks, e.g. in table cells (see Chapter 4.4.12 page 44)
- `mapattributes` Use attributes from the Word document, e.g. Attributes in tables (see Chapter 4.4.13 page 45)
- `value` Transfer a value to a FirstSpirit attribute, e.g. in the case of using inline tables the name of the table format template, or a fallback list type when converting lists from Word (see Chapter 4.4.14 page 46)
- `mapvalues` Convert list types from the Word document (see Chapter 4.4.15 page 48)
- `class` Convert self-defined Word style sheets, which are given the "class" attribute in the HTML clipboard (see Chapter 4.4.16 page 49)

Example:

```
<element handler="object" maptag="link" tag="a">
```

This `element` tag contains the assignment for links: these are marked up with the `a` tag in the HTML of the Word document. `maptag="link"` must be used in the ruleset to give links. Internally, `link` is reproduced on `CMS_LINK`.

4.3.5 attribute

The `attribute` tag is used to define the conversion rules for the HTML attributes of the Word document, e.g. for



- HTML style sheet definitions: `style`
- the definition of links: `href`
- the definition of tables: `colspan`, `rowspan`, `style`
- the integration of graphic pictures: `src`

An `attribute` tag **must** be specified in the XML definition for each HTML style sheet attribute contained in the Word document. It must be enclosed by `<element>... </element>` and therefore inherits all the attributes of the higher level element (see Chapter 4.2 page 21).

The following **mandatory attributes** must be specified for the `attribute` tag:

- `name` / `mapname`: These attributes are used to establish the link between the HTML information from the Word document and the FirstSpirit format and link templates: `name` is used to specify the values of the HTML attributes from the Word document which are to be converted on being imported into FirstSpirit; `mapname` gives the FirstSpirit value (abbreviation of the FirstSpirit format template or other values which are used internally to control FirstSpirit), into which the HTML attribute is to be translated. `mapname` does not need to be specified if the same name is used as that specified for the `name` attribute (see also Chapter 4.4.1 page 35 and Chapter 4.4.2 page 35).
- `handler`: This attribute defines how the HTML information from the Word document is to be processed (handled) (see also Chapter 4.4.5 page 37). In addition, `handler="style"` can also be used to import the information from the `style` tag (see Chapter 4.3.3 page 28).

Example: `<attribute handler="style" name="style"/>`

Example for **Links**:

```
<element handler="object" maptag="link" tag="a">
    <attribute mapname="target" name="href"/>
</element>
```

This `attribute` tag contains the assignment for the `href` attribute, with which a link's URL is specified. `mapname="target"` must be specified in the ruleset.

For further information on importing links, see Chapter 6.5 page 76.



Example for **pictures**:

```
<element content="IGNORE" handler="media" mediaref="src" tag="img">
    <attribute name="src"/>
</element>
```

This `attribute` tag is used to import the corresponding picture into the Media Store. `mapname="src"` must be specified in the ruleset; however, it can also be omitted, as the value of `mapname` and `name` is identical.

For further information on importing pictures, see Chapter 6.5 page 76.

The following **optional attributes** can be used with `attribute`:

- `mapvalues`: Converting list types from the Word document
Transferring a value to a FirstSpirit attribute, e.g., the name of the table format template when using inline tables, or a fallback list type when converting lists from Word (see Chapter 4.4.15, page 48)
- `value`: Transferring a value to a FirstSpirit attribute, e.g., the name of the table format template when using inline tables, or a fallback list type when converting lists from Word (see Chapter 4.4.14, page 46)

4.3.6 text

The `text` tag can be used to define how texts within elements are to be processed on being imported into the DOM input component.

The `text` tag must be enclosed by `<element>... </element>` and therefore inherits all the attributes of the higher level element (see Chapter 4.2 page 21).

The following **mandatory attribute** must be specified for the `text` tag:

`action`: This attribute is used to define how text is to be processed on being imported into a DOM input component.



The following values can be used for `action`:

Attribute	Explanation / Example	Mandatory attribute
ignore	<p>Text is not imported into the input component, i.e. it is ignored, e.g.</p> <pre data-bbox="553 541 886 569"><text action="ignore"/></pre>	no
default	<p>If a default element is defined (see Chapter 4.4.6 page 39), the text is converted with the attributes and values defined by this element, e.g.</p> <pre data-bbox="553 810 899 837"><text action="default"/></pre> <p>Line breaks (<code>
</code>) and spaces (<code>&nbsp;</code>) are removed and are not imported into the DOM input component.</p>	no
keep	<p>The text is imported into the DOM Editor, e.g.</p> <pre data-bbox="553 1152 857 1180"><text action="keep"/></pre> <p>This is the default setting.</p>	Default



4.4 Attributes

4.4.1 name

This mandatory attribute specifies the values of HTML attributes from the Word document which should be converted when importing to FirstSpirit (see Chapter 4.3.5, page 31).

4.4.2 mapname

This mandatory attribute specifies the FirstSpirit value (FirstSpirit format template tag or another value that is used internally for controlling FirstSpirit) into which the HTML attribute should be converted.

In addition to the FirstSpirit format template tags, the following values can be used:

- **Variable identifier:**

`align`, `bgcolor`, `color`: These values are variable identifiers for aligning text (`align`, for more information, see also Chapter 4.3.3, page 28), the background color (`bgcolor`), and the font color of text in a table cell (`color`), for example (for more information on this, see also *FirstSpirit Online Documentation*, Chapter "Template development" / "Template syntax" / "System objects" / "#style").

`style`: This value is the variable name for defining a list type (for more information, see also *FirstSpirit Online Documentation*, Chapter "Template development" / "Forms" / "Input components" / "DOM").

Examples:

```
<style mapname="align" name="text-align"/>
```

`style` in the context of unnumbered lists:

```
<attribute name="type" mapvalues="disc:1" mapname="style" value="0"/>
```



For restrictions on using lists and Word 2010, please see Chapter 6.4 page 73.

- **Keywords:**

`target`: `target` is the keyword for referencing links.



Example:

```
<attribute mapname="target" name="href"/>
```

`mapname` does not need to be specified if the same identifier is used as is specified for the `name` attribute (see Chapter 4.3.5, page 31).

4.4.3 tag

The `tag` attribute specifies the HTML element identifier in `element` tags, e.g.,

```
<element ... tag="ol"/>
```

for numbered lists.



For more information on the correct output of nested lists, see Chapter 5.5, page 63.

Additional possible tags are: `html`, `head`, `body`, `*`, `div`, `p`, `span`, `br`, `b`, `a`, `li`, `ul`, `table`, `tr`, `td`, `th`

4.4.4 mptag

The `mptag` attribute specifies the FirstSpirit format template tag in `element` tags that should be linked to the respective Word template, e.g.,

```
<element ... mptag="h1" .../>
```

for first-level headings in the Word document.

Additional tags may be: `br`, `b`, `pre`, `u`, `s`, `ul`, `td`.



The value `link` is used for links:

```
<element handler="object" mptag="link" tag="a">  
  <attribute mapname="target" name="href"/>  
</element>
```



If the `maptag` attribute is not specified, the same identifier as the one specified for the `tag` attribute is automatically used (see Chapter 4.4.3, page 36). If the format identifiers in FirstSpirit and in the Word document are identical, the `maptag` attribute does not need to be specified additionally.

4.4.5 handler

The mandatory `handler` attribute can be used to give different options, defining how the respective HTML element is to be handled on being imported into the DOM Editor. The `map` value is used as a default. In this case, the HTML Element is not converted from the Word document. It can be mapped with this tag name by giving the additional attribute `mapname`. If the `handler` attribute is not defined for an HTML element, `handler="map"` is automatically set.

The `skip` value can be used to skip the HTML element, i.e. it is ignored; `strip` removes the tag of the respective element on being imported into the DOM Editor (e.g. the `<html>` tag), but the content of the element is retained. If the `mapattributes="true"` attribute is also specified, the attributes of the element are applied to the higher level element (see Chapter 4.4.13 page 45).

The `default` value can be used to revert to the default conversion rules (see Chapter 4.4.6 page 39 and Chapter 4.5 page 50).

The values `object`, `media` and `table` are required to import links, pictures and tables. As with the `map` value, the respective HTML element is not converted. The following tags and attributes are also required to import these objects:

- for links: `attribute` tag for `href` (see Chapter 4.3.5 page 31)
- for pictures: `content` (see Chapter 4.4.9 page 41), `mediaref` (see Chapter 4.4.10 page 41)
- for tables: `element` and `attribute` tags for the conversion of
 - **Table rows** (Word HTML tag `tr`): e.g. `<element tag="tr">`
 - **Table cells** (Word HTML tag `td`): e.g. `<element inherit="HTML.paragraph" tag="td">`
 - **header cells** (Word HTML tag `th`): e.g. `<element inherit="HTML.tablecell" maptag="td" tag="th"/>`
 - **Merges** (Word HTML tags `colspan` and `rowspan`): e.g. `<attribute name="colspan"/>` and `<attribute name="rowspan"/>`
 - for the use of **inline tables**: `value` attribute for indication of the relevant table format template: e.g. `<attribute mapname="style" value="REFERENCE_NAME_OF_THE_TABLE_FORMAT_TEMPLATE"/>`



The value **style** can be used to import the information from the `style` tag (see Chapter 4.3.3 page 28).

Value	Explanation / Example	Mandatory information
map	<p>the HTML element is not converted on import, e.g.</p> <pre><element default="true" handler="map" id="HTML.paragraph" tag="p"></pre> <p>This is the default setting.</p>	Default
skip	<p>the HTML element is skipped on import, e.g.</p> <pre><element handler="skip" tag="*/></pre>	no
strip	<p>the tag of the HTML element is removed and the content of the HTML element is imported into the DOM input component, e.g.</p> <pre><element handler="strip" tag="html"></pre> <p>If <code>mapattributes="true"</code> is also specified, the attributes of this element are transferred to the higher-level element.</p>	no
default	<p>Revert to the default element, e.g.</p> <pre><element handler="default" tag="*/></pre>	no
object	<p>required to import links, e.g.</p> <pre><element handler="object" maptag="link" tag="a"></pre>	no



Value	Explanation / Example	Mandatory information
media	required to import media, e.g. <pre><element content="IGNORE" handler="media" mediaref="src" tag="img"></pre>	no
table	required to import tables, e.g. <pre><element handler="table" tag="table"></pre>	no
find	Read out tags from the Word document, e.g. <pre><element findtag="title" handler="find" tag="head"/></pre>	no
style	Import information from the <code>style</code> tag (see Chapter 4.3.3 page 28)	no

4.4.6 default

The `default` attribute can be used to define an element as a default conversion rule for unknown formatting ("default element"). If the value "true" is set for `default`, the conversion is used for all formatting or objects from the Word document for which there is no ruleset, e.g.

```
<element tag="p" handler="map" default="true" id="HTML.paragraph">
```

This rule can then be reverted to with `<element tag="*" handler="default">`. With this definition, all "unknown" HTML elements from the Word document are converted using the default conversion rule (see Chapter 4.5 page 50).

In the example given, unknown formatting and mark-ups in the Word document are transferred in a paragraph (`<p> . . . </p>`).



Only one default element should be defined for each ruleset. If several default elements are defined, the last element of the XML ruleset is always used.



Value	Explanation / Example	Mandatory information
true	Configuration of the <code>element</code> tag as the default conversion rule for unknown HTML formatting from the Word document, e.g. <pre><element default="true" handler="map" id="HTML.paragraph" tag="p"></pre>	no

4.4.7 id

The `id` attribute can be used to give `element` tags a name. Other `element` tags reference to this name via the `inherit` attribute (see Chapter 4.4.11 page 41) and therefore import all attributes and values, e.g.

```
<element default="true" handler="map" id="HTML.paragraph" tag="p">
...
<element handler="table" tag="table">
  <element tag="tr">
    <attribute name="colspan"/>
    <attribute name="rowspan"/>
    <attribute handler="style" name="style"/>
    <element inherit="HTML.paragraph" tag="td">
      <element handler="strip" inherit="HTML.paragraph"
        mapattributes="true" tag="p"/>
    </element>
  </element>
</element>
</element>
```

In this example, the element with the attribute `tag="p"` (continuous text paragraphs) is given the name "HTML.paragraph". Table cells (HTML tag `td`) and paragraphs within a table cell (HTML



tag `p` within the table, `tag="table"`) reference to this element via the `id` "HTML.paragraph" and therefore use values and attributes from continuous text paragraphs (see also Chapter 4.2 page 21).

4.4.8 findtag

The `findtag` attribute together with the `handler="find"` attribute (see Chapter 4.4.5 page 37) can be used to read out tags from the Word document, e.g.

```
<element findtag="title" handler="find" tag="head"/>
```

In this example, the text of the `title` element is read out of the `head` section of the Word document.

4.4.9 content

The `content` attribute is required to import pictures, e.g.

```
<element content="IGNORE" handler="media" mediaref="src" tag="img">
```

The `IGNORE` value ensures that the picture is not parsed.

4.4.10 mediaref

The `mediaref` attribute is required to import pictures, e.g.

```
<element content="IGNORE" handler="media" mediaref="src" tag="img">
```

`src` must always be specified as the value for `mediaref`.

4.4.11 inherit

The `inherit` attribute can be used by an `element` tag to inherit the values and attributes of another `element` tag. This must be defined by means of the attribute `id` (Chapter 4.4.7 page 40), e.g.



```
<element default="true" handler="map" id="HTML.paragraph" tag="p">
...
<element handler="table" tag="table">
  <element tag="tr">
    <attribute name="colspan"/>
    <attribute name="rowspan"/>
    <attribute handler="style" name="style"/>

    <element inherit="HTML.paragraph" tag="td">

      <element handler="strip" inherit="HTML.paragraph"
        mapattributes="true" tag="p"/>
    </element>

  </element>

</element>

</element>
```

In this example, the element with the attribute `tag="p"` (continuous text paragraphs) is given the name "HTML.paragraph". Table cells (HTML tag `td`) and paragraphs within a table cell (HTML tag `p` within the table, `tag="table"`) reference to this element via the `id` "HTML.paragraph" and therefore use values and attributes from continuous text paragraphs (see also Chapter 4.2 page 21).

The value `NONE` can be used to inhibit the respective `element` tag from inheriting the attributes and values of a higher-level `element` tag:



```
<element handler="table" tag="table" inherit="NONE">
  <element tag="*" handler="skip" />
  <element tag="tr">
    <element tag="td" id="HTML.tablecell"
      inherit="HTML.paragraph" >
      <attribute name="colspan"/>
      <attribute name="rowspan"/>
      <attribute handler="style" name="style"/>
      <element tag="p" handler="strip" mapattributes="true"
        break="br" inherit="HTML.paragraph"/>
    </element>
    <element tag="th" inherit="HTML.tablecell"
      maptag="td" />
  </element>
</element>
```

This example contains the rules for converting a table. `inherit="NONE"` is used to specify that all previously defined conversion rules are not applied to tables.



4.4.12 break

The `break` attribute can be used to define the conversion of text breaks, e.g.

```
<element handler="table" tag="table" inherit="NONE">
  <element tag="*" handler="skip" />
  <element tag="tr">
    <element tag="td" id="HTML.tablecell"
      inherit="HTML.paragraph" >
      <attribute name="colspan"/>
      <attribute name="rowspan"/>
      <attribute handler="style" name="style"/>
      <element tag="p" handler="strip" mapattributes="true"
        break="br" inherit="HTML.paragraph"/>
      </element>
      <element tag="th" inherit="HTML.tablecell"
        maptag="td" />
    </element>
  </element>
</element>
```

This example contains the rules for converting a table. `break="br"` is used to realise text breaks within table cells in Word in the DOM input component too, while `<p>` tags are removed (`tag="p" handler="strip"`).



4.4.13 mapattributes

The `mapattributes` attribute can be used to import attributes from the Word document into FirstSpirit, e.g.

```
<element handler="table" tag="table" inherit="NONE">
  <element tag="*" handler="skip" />
  <element tag="tr">
    <element tag="td" id="HTML.tablecell"
      inherit="HTML.paragraph" >
      <attribute name="colspan"/>
      <attribute name="rowspan"/>
      <attribute handler="style" name="style"/>
      <element tag="p" handler="strip" mapattributes="true"
        break="br" inherit="HTML.paragraph"/>
    </element>
    <element tag="th" inherit="HTML.tablecell"
      maptag="td" />
  </element>
</element>
```


This example contains the rules for converting a table. `mapattributes="true"` specifies that the attributes of the table from the Word document, e.g. `valign`, `bgcolor`, etc. are used in a DOM input component on import. The attributes are not used if the value `false` is set.



4.4.14 value

The `value` attribute can transfer a value to a FirstSpirit attribute. This currently has two applications:

4.4.14.1 Inline tables

If Word content is to be transferred to inline tables (DOM editor with parameter `table="yes"`), the reference name of the table format template used in the project that should be used for converting the tables from Word must be specified by `value`. Formatting resulting from the table format template is also considered during import in addition to formatting from the Word document. Cells from the Word document that have no background color, font color, and/or text alignment formatting are formatted according to the table format template specifications (if available). Otherwise, the cell properties cannot be edited and the  icon is not active.

Example:

```
<element handler="table" tag="table" inherit="HTML.body">
  <attribute mapname="style"
            value="REFERENCENAME_TABLEFORMATTEMPLATE" />
```

The `table` tag in the DOM editor then receives a `style` attribute with the value of the desired table format template:


```
<DOM>
...
  <table style="REFERENCENAME_TABLEFORMATTEMPLATE">
...
  </table>
...
</DOM>
```





The reference name of the table format template must be adapted to be relevant project in the sets of rules that were supplied. For example, the reference name of the table format template for the "Mithras Energy" demo project is "textpicture".



The properties of the tables imported from Word (e.g., cell background color, font color, alignment) can only be edited further in FirstSpirit if these properties are also defined in the underlying table format template for the cells concerned (display rules). Otherwise, clicking on the  icon triggers a corresponding error message and the cell properties cannot be edited.

If only individual table cells in the Word document are formatted, you could remove this formatting before import if it is not available in the table format template. Once imported, the cells can then be formatted using the options available in the DOM editor.

If the Word document contains tables that are based on a standard template (e.g., the top or bottom row or every second row or column contrasts with the other rows/columns), this template can be applied using the table format template (incl. display rules) and the individual formatting can be suppressed by the editor in the DOM editor by using `hidden="yes"`, for example.

4.4.14.2 Lists

When defining rules for lists, `value` can be used to define a list type that should be used as a fallback solution (e.g., if no list type has been defined in FirstSpirit for a list type in the Word document; for more information, see also Chapter 4.4.15, page 48).

Example:

```
<element tag="ul" id="HTML.list">
  <attribute name="type" mapvalues="disc:1"
    mapname="style" value="0"/>
  <element tag="li" inherit="HTML.paragraph"
    mapattributes="false"/>
</element>
```



In this example, a dash would be used as the fallback value for all Word list symbols in unnumbered lists (`tag="ul"`) apart from `disc`.



For restrictions on using lists and Word 2010, please see Chapter 6.4 page 73.

4.4.15 mapvalues

The `mapvalues` attribute defines how lists from Word are displayed in FirstSpirit. The values used in Word must be separated from the FirstSpirit list types using a colon (":"). Multiple value pairs are separated by a comma.

In **unnumbered lists** (`ul`), the value for the list symbol from the Word document is specified before the colon (e.g., `disc`, `circle`, `square`), the required FirstSpirit list type is specified after the colon, e.g.,

```
<attribute mapname="style" mapvalues="disc:1" name="type"
value="0"/>
```

In **numbered lists** (`ol`), the numbering style from the Word document is specified before the colon (`1`, `I`, `i`, `A`, `a`), the required FirstSpirit list type is specified after the colon, e.g.,

```
<attribute mapname="style" mapvalues="1:2,a:3,A:4,i:5,I:6"
name="type" value="0"/>
```

The FirstSpirit list types are as follows:

- 0: Dash (-)
- 1: Bullet
- 2: Numbered
- 3: Alphabetical, lower case
- 4: Alphabetical, upper case
- 5: Roman numerals, lower case
- 6: Roman numerals, upper case
- 7: Empty/Indents only
- 8: User-defined





When defining the display of Word list types to FirstSpirit list types, you should consider that the FirstSpirit types 0 to 8 listed above can be displayed regardless of which list types are specified by the parameter `listConfig`. However, only the list types defined by the parameter `listConfig` can be used in the DOM editor to edit lists further. For more information on list types, see *FirstSpirit Online Documentation*, section "Template development" / "Template syntax" / "System objects" / "#list", call "#list.style".



`value` can also be used to specify which FirstSpirit list type should be defined as the fallback value. If `value="0"`, a dash would be used as the fallback value, for example (see also Chapter 4.4.14, page 46).



For restrictions on using lists and Word 2010, please see Chapter 6.4 page 73.

4.4.16 class

The `class` attribute is used to map classes from the Word document with format or link templates in FirstSpirit, e.g.

```
<element class="Wichtig" maptag="important" tag="p"/>
```

In this example, the Word character style sheet "Wichtig" is mapped with the FirstSpirit format template with the abbreviation "important", i.e. paragraphs formatted in the Word document using the paragraph style sheet "Wichtig" are formatted with the character format template "Important" on being imported into the DOM Editor.



4.5 Handling unknown tags

Format or paragraph style sheets from the Word document, which are not defined in the XML rulesets cannot be accordingly converted in FirstSpirit.

A default response can be defined for this case, e.g. that all unknown formatting from Word is converted in a paragraph (<p>) in FirstSpirit.

To do this, the `handler` attribute in the `element` tag for continuous text paragraphs is set to the value "default" (see Chapter 4.4.6 page 39), e.g.

```
<element default="true" handler="map" id="HTML.paragraph" tag="p">
```

An `element` tag with `tag="*"` is created to format "unknown" elements from Word with the "paragraph formatting" defined above. `*` acts as a wildcard.

```
<element handler="default" tag="*" />
```

4.6 Example: Creating a rule to reproduce an individual Word format style sheet

This chapter explains the procedure for creating your own XML rules, using the example of the Word style sheet "Hinweis" (German for "Note").

4.6.1 Step 1: Preparing the Word document

The Word document contains the "Hinweis" style sheet. It has the following properties: red font colour (RGB colour 255, 0, 0) and bold. A paragraph is formatted with this style sheet. In the Word test document e.g.

This text is formatted with the paragraph format "Hinweis", with a font colour with RGB value 255, 0, 0."

4.6.2 Step 2: Creating the necessary format template in FirstSpirit

A format template is required in FirstSpirit, which can reproduce the formatting of the Word "Hinweis" style sheet in the DOM Editor. It should also be red and bold.

A format template with the following properties is created for this purpose:



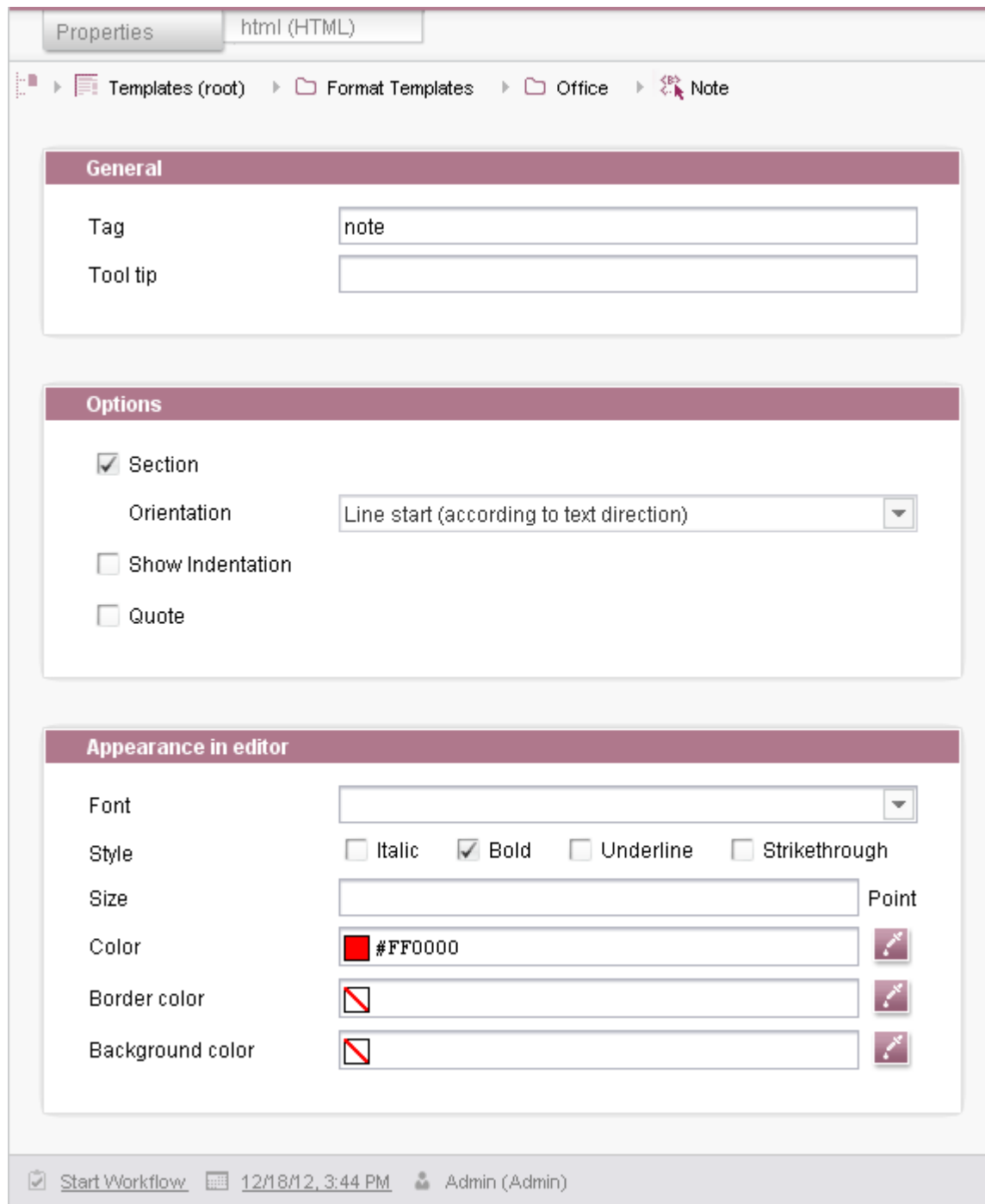


Figure 4-2: "note" format template

In addition, the output channels also have to be configured accordingly.



4.6.3 Step 3: Outputting the Word document in HTML

This paragraph is output as follows in the HTML format of the Word document:

```
<p class="Hinweis">This text is formatted with the paragraph format "Hinweis", with the font colour RGB 255, 0, 0. </p>
```

4.6.4 Step 4: Analysis of the HTML

The `<p>` indicates that it is a paragraph, `class` specifies the name of the style sheet.

4.6.5 Step 5: Creating the XML rule definition

An `element` tag is required for the rule definition. As this is an additional rule description for a paragraph, the tag must be located on the level below the rule description for the HTML tag "body". For the `<p>` from the HTML, the XML requires a `tag` attribute:

```
tag="p"
```

However, all properties defined in the XML for paragraphs should also apply to this format template. This is implemented with the `inherit` attribute:

```
inherit="HTML.paragraph"
```

`HTML.paragraph` is the ID of the element (in this case the rule description for paragraphs), whose values and attributes are to be imported.

As this is a self-defined Word style sheet, it is defined with `class`. The `class` attribute with which the name of the Word style sheet is given must therefore be used in the XML ruleset:

```
class="Hinweis"
```

The `maptag` attribute is imported into the corresponding FirstSpirit format template:

```
maptag="note"
```



4.7 Default ruleset of the FirstSpirit OfficeConnect module

The following XML ruleset is supplied as a standard component with the OfficeConnect module:

```
<?xml version="1.0" encoding="UTF-8"?>
<ImportRuleSets>
  <mapping description="Project local import ruleset definition"
linkConfigExternal="textlinkexternal" linkConfigInternal="textlinkinternal"
mimeType="text/html" name="Project local Import Ruleset" versionTag="22">

  <style mapname="align" name="text-align"/>
  <style mapname="foregroundcolor" name="color"/>
  <style mapname="bgcolor" name="background"/>
  <style name="list-style"/>

  <element handler="strip" tag="html">
    <attribute handler="style" name="style"/>
    <text action="ignore"/>
    <element tag="head" handler="find" findtag="title" />
    <element handler="strip" id="HTML.body" tag="body">

      <text action="default" />
      <element tag="*" handler="default" />
      <element tag="div" handler="strip" />
      <element tag="p" class="Wichtig" maptag="important"
        inherit="HTML.paragraph" />
      <element tag="p" class="Hinweis" maptag="note"
        inherit="HTML.paragraph" />

      <element default="true" handler="map" id="HTML.paragraph"
        tag="p">
        <text action="keep" />
        <element tag="*" handler="strip" />
        <element tag="span" handler="strip" />

        <element content="IGNORE" handler="media" mediaref="src"
          tag="img">
          <attribute name="src"/>
        </element>

        <element maptag="br" tag="br"/>
        <element maptag="b" tag="b"/>
        <element tag="strong" maptag="b" />
        <element maptag="i" tag="i"/>
        <element maptag="pre" tag="pre"/>
        <element maptag="u" tag="u"/>
        <element maptag="s" tag="s"/>

        <element maptag="h1" tag="h1"/>
        <element maptag="h2" tag="h2"/>
        <element maptag="h3" tag="h3"/>
        <element maptag="h4" tag="h4"/>
        <element maptag="h5" tag="h5"/>
      </element>
    </element>
  </element>
</mapping>
</ImportRuleSets>
```



```

        <element tag="p" class="Wichtig" maptag="important"
            inherit="HTML.paragraph" />
        <element tag="p" class="Hinweis" maptag="note"
            inherit="HTML.paragraph" />

        <element handler="object" maptag="link" tag="a">
            <attribute mapname="target" name="href"/>
        </element>

        <element tag="ul" id="HTML.list">
            <element inherit="HTML.paragraph" tag="li"
                mapattributes="false"/>
        </element>

        <element tag="ol" maptag="ul" inherit="HTML.list" />

        <element handler="table" tag="table" inherit="NONE">
            <element tag="*" handler="skip" />
            <element tag="tr">
                <element tag="td" id="HTML.tablecell"
                    inherit="HTML.paragraph" >
                    <attribute name="colspan"/>
                    <attribute name="rowspan"/>
                    <attribute handler="style" name="style"/>
                    <element tag="p" handler="strip"
                        mapattributes="true" break="br"
                        inherit="HTML.paragraph"/>
                </element>
                <element tag="th" inherit="HTML.tablecell"
                    maptag="td" />
            </element>
        </element>
    </element>
</mapping>
<mapping description="Project local import ruleset definition (generic
link)" linkConfigExternal="textlinkexternal"
linkConfigInternal="textlinkinternal" mimeType="text/html" name="Project local
Import Ruleset (generic link)" versionTag="22">

    <style mapname="align" name="text-align"/>
    <style mapname="foregroundcolor" name="color"/>
    <style mapname="bgcolor" name="background"/>
    <style name="list-style"/>

    <element handler="strip" tag="html">
        <attribute handler="style" name="style"/>
        <text action="ignore"/>
        <element tag="head" handler="find" findtag="title" />
        <element handler="strip" id="HTML.body" tag="body">

            <text action="default" />

```



```
<element tag="*" handler="default" />
<element tag="div" handler="strip" />
<element tag="p" class="Wichtig" maptag="important"
  inherit="HTML.paragraph" />
<element tag="p" class="Hinweis" maptag="note"
inherit="HTML.paragraph" />

<element default="true" handler="map" id="HTML.paragraph"
tag="p">
  <text action="keep" />
  <element tag="*" handler="strip" />
  <element tag="span" handler="strip" />

  <element content="IGNORE" handler="media" mediaref="src"
    tag="img">
    <attribute name="src"/>
  </element>

  <element maptag="br" tag="br"/>
  <element maptag="b" tag="b"/>
  <element tag="strong" maptag="b" />
  <element maptag="i" tag="i"/>
  <element maptag="pre" tag="pre"/>
    <element maptag="u" tag="u"/>
    <element maptag="s" tag="s"/>

  <element maptag="h1" tag="h1"/>
  <element maptag="h2" tag="h2"/>
  <element maptag="h3" tag="h3"/>
  <element maptag="h4" tag="h4"/>
  <element maptag="h5" tag="h5"/>

  <element tag="p" class="Wichtig" maptag="important"
    inherit="HTML.paragraph" />
  <element tag="p" class="Hinweis" maptag="note"
inherit="HTML.paragraph" />

  <element handler="object" maptag="link" tag="a">
    <attribute mapname="target" name="href"/>
  </element>

  <element tag="ul" id="HTML.list">
    <element inherit="HTML.paragraph" tag="li"/>
  </element>

  <element tag="ol" maptag="ul" inherit="HTML.list" />

  <element handler="table" tag="table" inherit="NONE">
    <element tag="*" handler="skip" />
    <element tag="tr">
      <element tag="td" id="HTML.tablecell"
inherit="HTML.paragraph" >
        <attribute name="colspan"/>
        <attribute name="rowspan"/>
        <attribute handler="style" name="style"/>
      </element>
    </element>
  </element>

```




```
        <element tag="p" handler="strip"
            mapattributes="true" break="br"
            inherit="HTML.paragraph"/>
        </element>
        <element tag="th" inherit="HTML.tablecell"
            maptag="td" />
        </element>
    </element>
</element>
</element>
</mapping>
<mapping description="use a default text only import handler."
mimeType="text/plain" name="Standard (text only import)" versionTag="21"/>
</ImportRuleSets>
```



5 Configuring the Input Components

The OfficeConnect function can be used in the DOM Editor and in the DOM Table.



Only tables from Word documents can be imported into the DOM Table input component, not continuous texts (see also Chapter 6.7 page 81).

5.1 Activating the import function

In order to make the Office module function available in the DOM Editor or in the DOM table, the parameter `enableImport="yes"` must be added to the required section, page or table templates in the form area of the `CMS_INPUT_DOM` or `CMS_INPUT_DOMTABLE` input component. The input components are then assigned an import icon when they are integrated in the Page or Content Store:

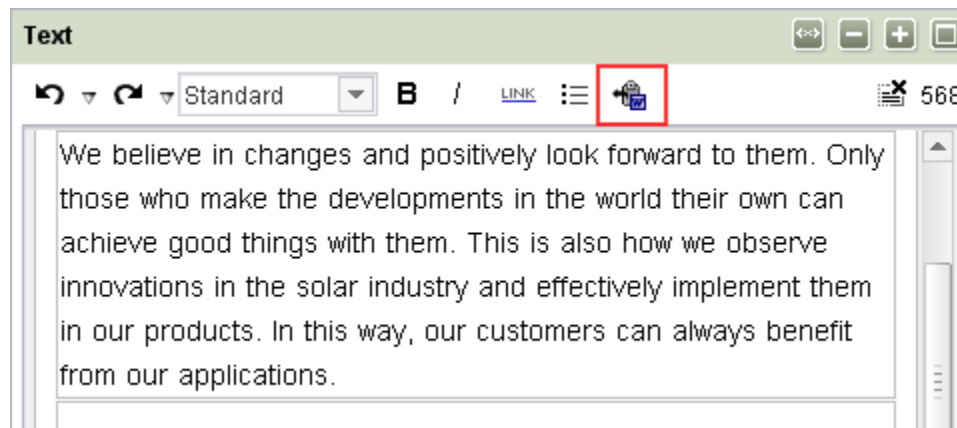
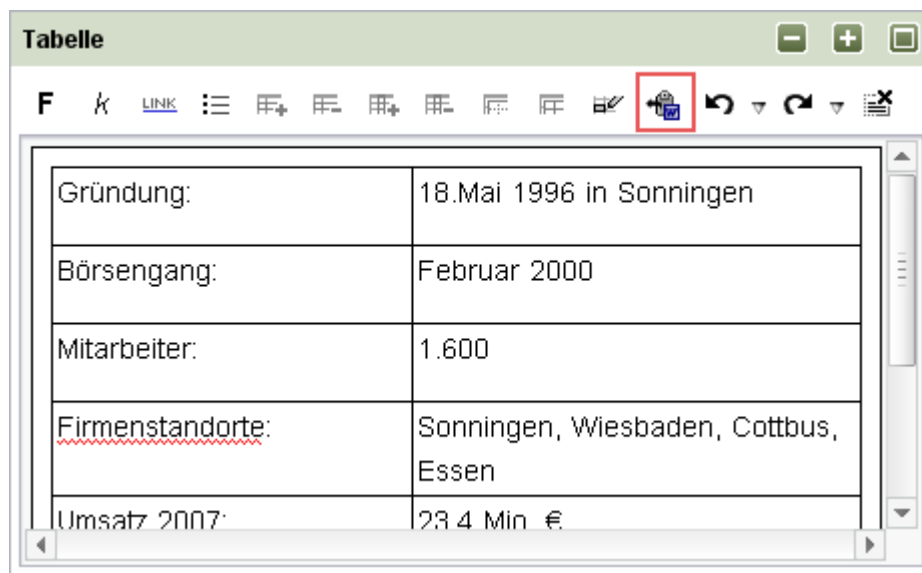


Figure 5-1: DOM Editor with icon for the Office import





Gründung:	18.Mai 1996 in Sonningen
Börsengang:	Februar 2000
Mitarbeiter:	1.600
Firmenstandorte:	Sonningen, Wiesbaden, Cottbus, Essen
Umsatz 2007:	23.4 Min. €

Figure 5-2: DOM Table with icon for the Office import

5.2 Limiting selectable rulesets

The optional parameter `importRuleset` can be used to define which rulesets are to be made available to the respective DOM Editor or DOM Table and can therefore be selected by the editor. This is done by giving the name of the ruleset defined in the XML definition (see Chapter 4.3.2 page 25) in double inverted commas. These can be both rulesets defined for the whole server and those defined for the respective project only, e.g.:

```
importRuleset="Project"
```

In this case, only the ruleset named "Project" is made available.

More than one ruleset can be specified. In that case, the rulesets must be given separated by a comma.

If the parameter is not specified, all rulesets valid for the respective project and the rulesets valid server-wide are displayed for selection in the input component.



5.3 Restrictions (format and link templates, lists, tables)

5.3.1 Format templates

In order for the assignment between the content from the Word document and the FirstSpirit format templates to work properly, these format templates must not be restricted in the required DOM Editor or DOM Table.

The use of format templates can be restricted in the DOM-Editor by using the `<FORMATS>` tag. Each format template which is to be admitted within the respective input component is given in a separate `<TEMPLATE>` tag between the opening and closing `<FORMATS>` tag:

```
<FORMATS>
  <TEMPLATE name="Tag of the format template">
</FORMATS>
```

If no `<FORMATS>` tag is specified all format templates are allowed.

If a formatting, e.g. **bold** is used in the Word document, also the FirstSpirit format template for **bold** (standard format template with the tag "b") should be allowed for the DOM-Editor. Otherwise, content formatted in bold can be imported into the DOM-Editor and the also format will be displayed, but the format cannot be changed in the DOM-Editor. The icon for the format **bold** is deactivated in this case.

If the use of format templates is restricted for the DOM-Editor, to which the Word document is to be imported, pay attention that all formats used in Word are admitted in the DOM-Editor.

The same applies to links, lists and tables:

5.3.2 Link templates

The use of links can be restricted in the DOM-Editor by using the `<LINKEDITORS>` tag. Each link template which is to be admitted within the respective input component is given in a separate `<LINKEDITOR>` tag between the opening and closing `<LINKEDITORS>` tag, e.g.:

```
<LINKEDITORS>
  <LINKEDITOR name="Unique name of the link template">
</LINKEDITORS>
```

If no `<LINKEDITORS>` tag is specified all link templates are allowed.



If the use of link templates is restricted for the DOM-Editor, to which the Word document is to be imported, pay attention that the required link templates (for the use of the standard rulesets "textlinkinternal" and "textlinkexternal") are admitted in the DOM-Editor.

5.3.3 Lists

The use of lists can be restricted in the DOM-Editor using the parameter `list="NO"`. If lists are used in the Word document, which are to be imported into the DOM-Editor and edited there, pay attention that the parameter is not set to `NO`. Depending on the project configuration and the Word document the list configuration must be possibly changed, too (attributes `listConfig` and `listDefaultConfig`).



For notes about the correct output of nested lists see also Chapter 5.5 page 63.



For restrictions on using lists and Word 2010, please see Chapter 6.4 page 73.

5.3.4 Tables (only DOM-Editor)

The use of tables in the DOM-Editor (so-called "Inline tables") can be controlled using the attribute `table`. If tables are used in the Word document the table mode should be activated for the respective DOM-Editor (`table="YES"`).

For further information about the configuration of the DOM-Editor and the DOM-Table see FirstSpirit Online Documentation, section "Template development" / "Forms" / "Input components" / "DOM" or "DOMTABLE".



5.4 Using link templates

Link templates are used for transferring external links and images from Word documents. When using link templates, you must ensure that the special variable identifiers for the input components that determine how information from the Word document should be displayed are used. These input components are defined via the form tab. In addition, on the "Properties" tab, settings can be made to determine which input component should be used for displaying link text and link images in the DOM editor:

Intended purpose	Function Information from the Word document Usable input components	Variable identifier in FirstSpirit
External links	Transfer of the link target (external URL) to Microsoft Word: "Address" field Incorporation into FirstSpirit via CMS_INPUT_TEXT	ref
	Transfer of the text that should be linked to Microsoft Word: "Display text as" field Incorporation into FirstSpirit via CMS_INPUT_TEXT	text
Images	Transfer of the image from the Word document Incorporation into FirstSpirit via FS_REFERENCE or CMS_INPUT_PICTURE	mediaref



Example of the link template form for incorporating images and external links:

```
<CMS_MODULE>
  <CMS_INPUT_TEXT name="ref">
    <LANGINFOS>
      <LANGINFO lang="*" label="Link target"/>
    </LANGINFOS>
  </CMS_INPUT_TEXT>

  <CMS_INPUT_TEXT name="text">
    <LANGINFOS>
      <LANGINFO lang="*" label="Link text"/>
    </LANGINFOS>
  </CMS_INPUT_TEXT>

  <FS_REFERENCE name="mediaref" imagePreview="yes">

    <FILTER>

      <ALLOW type="picture"/>

      <ALLOW type="file"/>

    </FILTER>

    <LANGINFOS>

      <LANGINFO lang="*" label="Link image"/>

    </LANGINFOS>

    <PROJECTS>

      <LOCAL name=".">

        <SOURCES>

          <FOLDER name="root" store="mediastore"/>

        </SOURCES>

      </LOCAL>

    </PROJECTS>

  </FS_REFERENCE>
</CMS_MODULE>
```



5.5 Output of lists

If nested lists are used in Microsoft Office Word, their representation in HTML differs from the recommendations of the World Wide Web Consortiums (W3C).

Example HTML code, generated from Word:

```
<ul>
  <li>abc</li>
  <ol>
    <li>bcd</li>
  </ol>
  <li>cde</li>
</ul>
```

According to the HTML 4.01 specification of the W3C only LI elements are allowed for UL/OL elements (see <http://www.w3.org/TR/html401/struct/lists.html>).

When imported into a DOM editor a valid list will be created from nested list, made in Word, by inserting automatically LI elements:

```
<ul style="list-style-type: none;">
  <li>abc</li>
  <li><ul>
    <li>bcd</li>
  </ul></li>
  <li>cde</li>
</ul>
```



In FirstSpirit ordered lists ("ol") are saved as unordered lists ("ul")!



However, there are differences when output by the browser, as shown in the following figures: Figure 5-3 shows the output of the HTML code generated by Microsoft Word in the browser, Figure 5-4 shows the output of the HTML code generated by FirstSpirit in the browser:

- First level, first item
 1. Second level, first item
- First level, second item

Figure 5-3: Output of nested lists (Microsoft Word)

- First level, first item
 - Second level, first item
- First level, second item

Figure 5-4: Output of nested lists (FirstSpirit)

To adjust the render method of FirstSpirit in the browser to that of Word, the FirstSpirit standard format template "List entry" with the tag "li" must be edited.

For this purpose

```
<li>${CMS_VALUE(#content)}</li>
```

in the HTML channel must be replaced as follows:

```

${CMS_IF(!#listitem.element.firstChild.isNull &&
  #listitem.element.firstChild.nodeName == "ul")$

${CMS_VALUE(#content)}$

${CMS_ELSE}$

  <li>${CMS_VALUE(#content)}</li>

${CMS_END_IF}$

```



*If the standard format template "li" is changed in this way, this will have an effect on the output of **all** lists in FirstSpirit. To prevent this and to change the output only for DOM editors which are to work with nested lists from Word, accordingly adjusted format templates can be used. At least the format template "Standard" (for sections which are to be formatted with a <p> tag) should be adjusted, as the following example shows:*



For this purpose a format template with activated option "Section" is to be created. A variable (e.g. with the name `_isWordList`) must be defined in the HTML channel:

```
$CMS_SET(_isWordList, true)$  
  
  <p>$CMS_VALUE(#content)$</p>  
  
$CMS_SET(_isWordList, null)$
```

This format template must be available in the DOM editor, which is to use nested lists:

```
<FORMATS>  
  <TEMPLATE name="IDENTIFIER_FORMAT_TEMPLATE"/>  
</FORMATS>
```

The HTML channel of the standard format template "li" must be modified in this case as follows:

```
$CMS_IF(!_isWordList.isNull &&  
  _isWordList &&  
  !#listitem.element.firstChild.isNull &&  
  #listitem.element.firstChild.nodeName == "ul")$  
  
$CMS_VALUE(#content)$  
  
$CMS_ELSE$  
  
<li>$CMS_VALUE(#content)$</li>
```



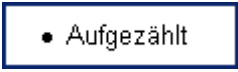
For restrictions on using lists and Word 2010, please see Chapter 6.4 page 73.

6 Editorial Work in SiteArchitect

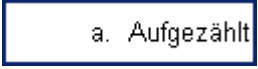

The following explanations primarily refer to the Word test document supplied with the module.

6.1 Explanations concerning the Word document

The following table lists the style sheets used in the Word document and gives their corresponding FirstSpirit templates (some are default format templates, some are project-specific templates from the "Mithras Energy" demo project):

Word style sheet	Example	FirstSpirit template
Überschrift 1	"Test document for the import from Word with FirstSpirit OFFICE"	h1
Überschrift 2	"Simple body text"	h2
Überschrift 3	"Body text with line breaks"	h3
Standard	"Section 1: Lorem ipsum dolor sit amet..."	p
Fett	"This sentence is completely bolded."	b
Kursiv	"This sentence is completely italicised."	k
Unterstrichen	"This sentence is completely underlined."	u
Aufgezählt (Symbol)	"• First bullet point" 	ul / li



Word style sheet	Example	FirstSpirit template
Aufgezählt (Letter) 	"a. Second level, first bullet point"	ul / li
Hinweis	"This text is formatted with the paragraph format "Hinweis", with the font colour RGB 255, 0, 0."	note
Wichtig	"3.4 (Paragraph formatting = "Wichtig")"	important
Hyperlink	" http://www.firstspirit.de ", " www.firstspirit.de ", "latest news"	external link template textlinkexternal
<Pictures>		internal link template textlinkinternal

The **paragraphs** "Section 1", "Section 2" and "Section 3" represent text sections defined in HTML by <p> tags. There is also an empty line between paragraphs 2 and 3. There is a soft line break in "Section 4", in front of the line "Consetetur sadipscing elitr...".

The FirstSpirit logo is integrated in the text body once as an **picture** (paragraph "pictures in lines"), and it is also present once as its own paragraph (paragraph "pictures in paragraphs"). In addition, the e-Spirit logo is integrated in a table cell ("Tables with merging and formatting").

The **character formatting** used are "Fett", "Kursiv", "Unterstrichen", "Durchgestrichen", in English: "bold", "italic", "underline", "struck through", and combinations of these. There default heading levels contained in Word are used as **paragraph formatting**. In addition, the self-defined Word paragraph style sheet "Hinweis" is used.

Links can be reproduced in the DOM Editor as an external link, irrespective of whether the link is identified as such in the Word document by formatting (blue font, underlined) (e.g. "<http://www.firstspirit.de>") or a link which is not identified as such (e.g. "[latest news](http://www.e-spirit.de)", linked to "http://www.e-spirit.de").

The XML set of rules supplied currently displays **bulleted lists** as filled-in circles (bullet points) or with lower case Roman numerals. However, only the list types "dash" and "Arabic numerals" are available in the DOM editor in the "Mithras Energy" demo project. Due to a limitation in the format when exporting from Word, FirstSpirit can only correctly display first and second level bulleted



lists. All subsequent levels are formatted as text, e.g., with spaces rather than tab indents.



For notes about the correct output of nested lists see also Chapter 5.5 page 63.

The Word format styles "Standard", "Fett", "Kursiv", "Unterstrichen" etc., in English "standard", "bold", "italics", "underline", etc. are also used in the **tables**. The cells are consecutively numbered, and the number in front of the dot indicates the column number and the number after the dot indicates the row number. For example, "2.3" stands for a cell in the second column of the third row. Merged cells are represented by a "+": "2.2 + 3.2" means, e.g., that the cell of the second column in the second row has been merged with the cell below it (second column, third row). There is an empty line between each of the tables and the preceding text.

6.2 Importing into FirstSpirit SiteArchitect

If the FirstSpirit OfficeConnect module is configured for a project and is activated in a DOM Editor or a DOM table, the "Import" button is visible in SiteArchitect:

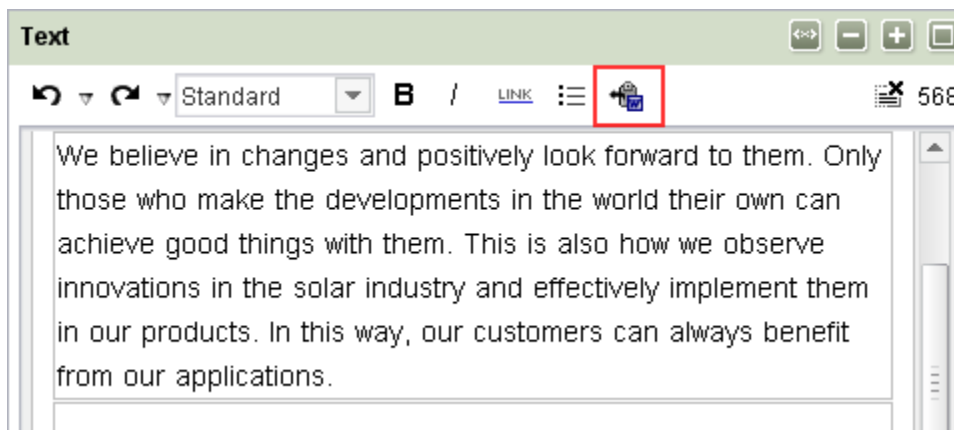


Figure 6-1: DOM Editor with Import button

In order to import content from Word into the DOM Editor, the cursor must first be placed in the position in which the content is to be inserted.

The content to be imported is then marked in the Word document and is copied onto the clipboard using <CTRL> + <C> or using the context menu (right-click mouse button). After clicking the Import button in the input component, or using the key shortcut <CTRL> + <V>, the content from the Word document is inserted (pasted) directly into the input component at the selected position and automatic formatting takes place in accordance with the ruleset specified



by the template developer.

If several rulesets are available for the input component, a window appears from which the ruleset by which the content is to be transformed can be selected:

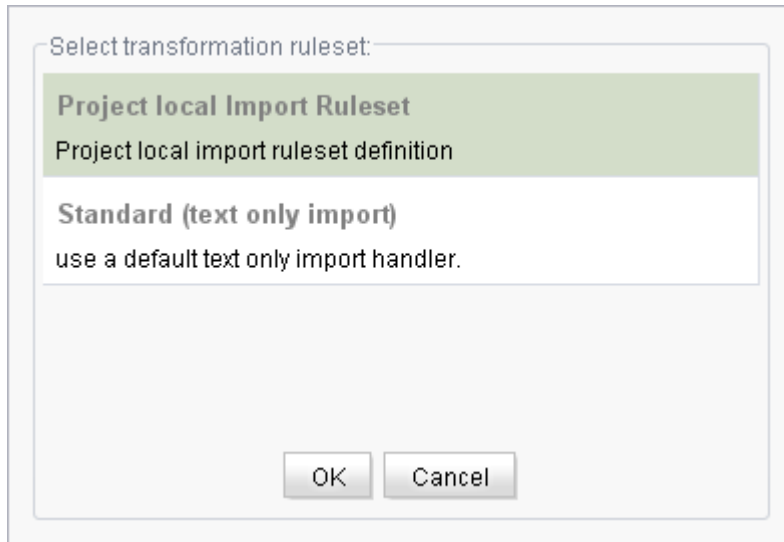


Figure 6-2: Select transformation ruleset

The required set of rules is marked using the mouse and the selection is confirmed by clicking . Alternatively, you can double-click on the set of rules. If a message is displayed stating "The link template of the import module defined in the server/project configuration could not be loaded. (textlinkinternal)", it means that the link template required for the import (in this case with the identifier "textlinkinternal") is not available. Repeat the import with another set of rules or contact the responsible project administrator or the template developer.

The content from the Word document is inserted into the input component in the chosen position and the formatting is applied according to the desired set of rules. If the dialog is closed by clicking , no text is inserted into the input component.

The content can then be edited as usual using the functions available in the input component.



If no Word format was copied onto the clipboard, after the Import button is clicked (or <CTRL> + <V>), a message appears "No compatible format found on the clipboard!" The import does not take place.





In order to ensure the OfficeConnect module functions properly, continuous text from Word documents should only be inserted into the DOM Editor or into so-called inline tables, tables from Word should only be inserted into the DOM Table or into so-called inline tables. Otherwise, the inserted content cannot be further edited.

6.3 Importing formatted texts

Continuous texts from a Word document can be imported into DOM input components along with their paragraph breaks, and character and paragraph formatting.

6.3.1 Text from Word documents

The following area is marked in the Word test document and is copied onto the clipboard (<CTRL> + <C>):



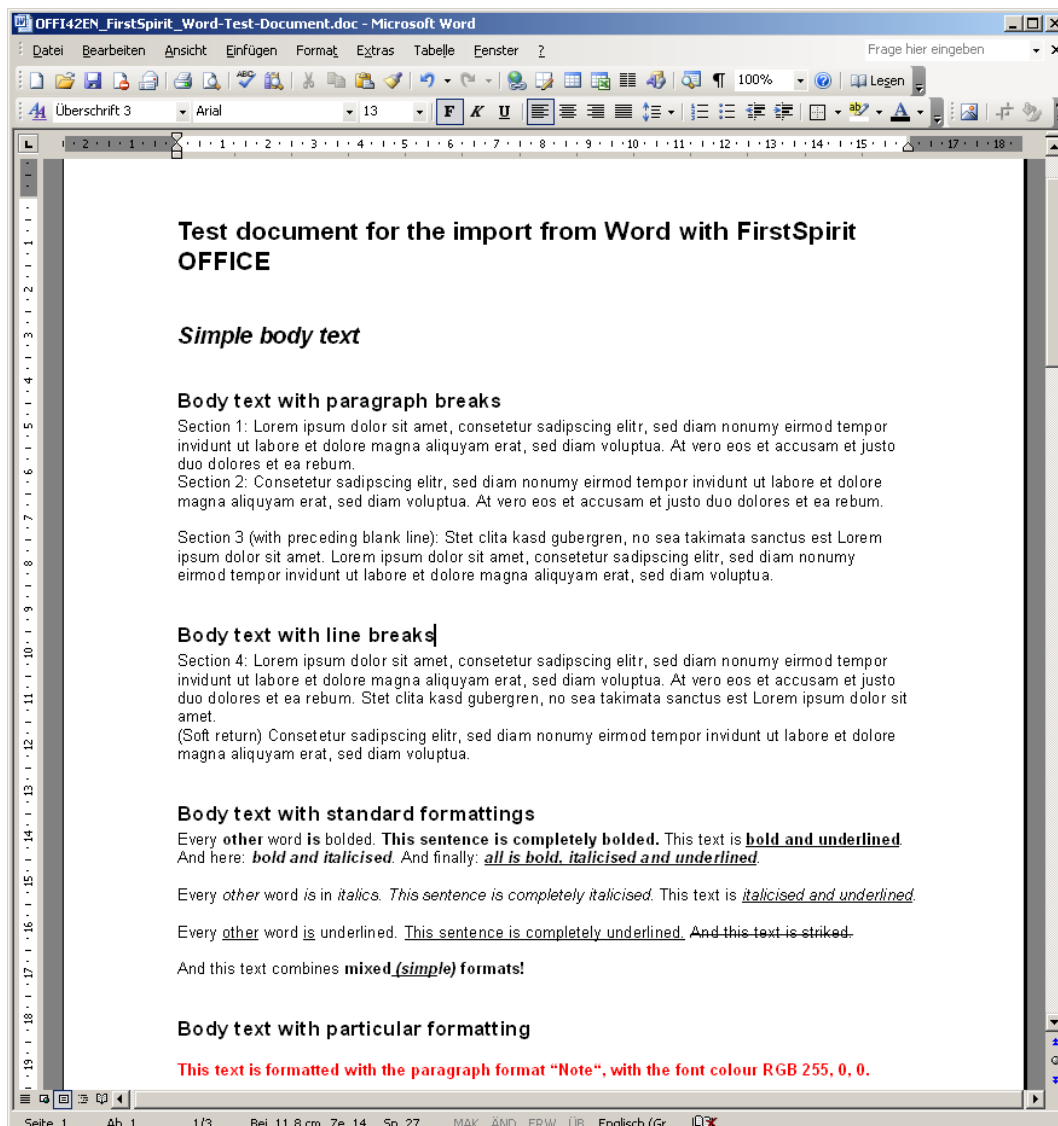


Figure 6-3: Word test document: Continuous text

6.3.2 Text in the DOM Editor

After it has been imported using the OfficeConnect function, the text from the test document can be displayed in the DOM Editor as follows:





Figure 6-4: DOM Editor with text from the Word document (DE)

Explanation: The headings of the first three levels are reproduced in FirstSpirit on the corresponding FirstSpirit format templates with the abbreviations (quicktags) "h1", "h2" and "h3". Paragraph 1 and paragraph 2 represent text sections, which are defined in HTML by `<p>...</p>`. They are reproduced in the DOM Editor in the default format template "Standard".



There is an empty line between paragraphs 2 and 3.

The character formatting from the Word document for bold, italics and underline are reproduced in the FirstSpirit format templates in the DOM Editor with the abbreviations (quicktags) "b", "i", "u" and "s". The "Hinweis" paragraph formatting from the Word document is reproduced on the format template in the DOM Editor with the abbreviation "note".

6.3.3 Exemplary HTML output

Depending on the template developer's definition, the HTML output on the website could look like the following:

```
<h1> Test document for the import from Word with FirstSpirit OFFICE </h1>
<h2>Simple body text</h2>
<h3>Body text with paragraph breaks</h3>
    <p class="section">Section 1: Lorem ipsum dolor sit
    amet, consetetur sadipscing elitr, sed diam nonumy
    eirmod tempor invidunt ut labore et dolore magna
    aliquyam erat, sed diam voluptua. At vero eos et accusam
    et justo duo dolores et ea rebum.</p>
    ...
```

6.4 Importing lists

Due to limiting in the export format, lists from Word are only imported up to their second level when imported into DOM input components.



*There are some limitations when transferring bulleted lists from **Word 2010**:*

- *The compatibility option "Use default format templates for bulleted lists or numbered lists" must be activated.*
- *The list must not be outdented to the left.*





For notes about the correct output of nested lists see also Chapter 5.5 page 63.

6.4.1 Lists from Word documents

The following area is marked in the Word test document and is copied onto the clipboard (<CTRL> + <C>):

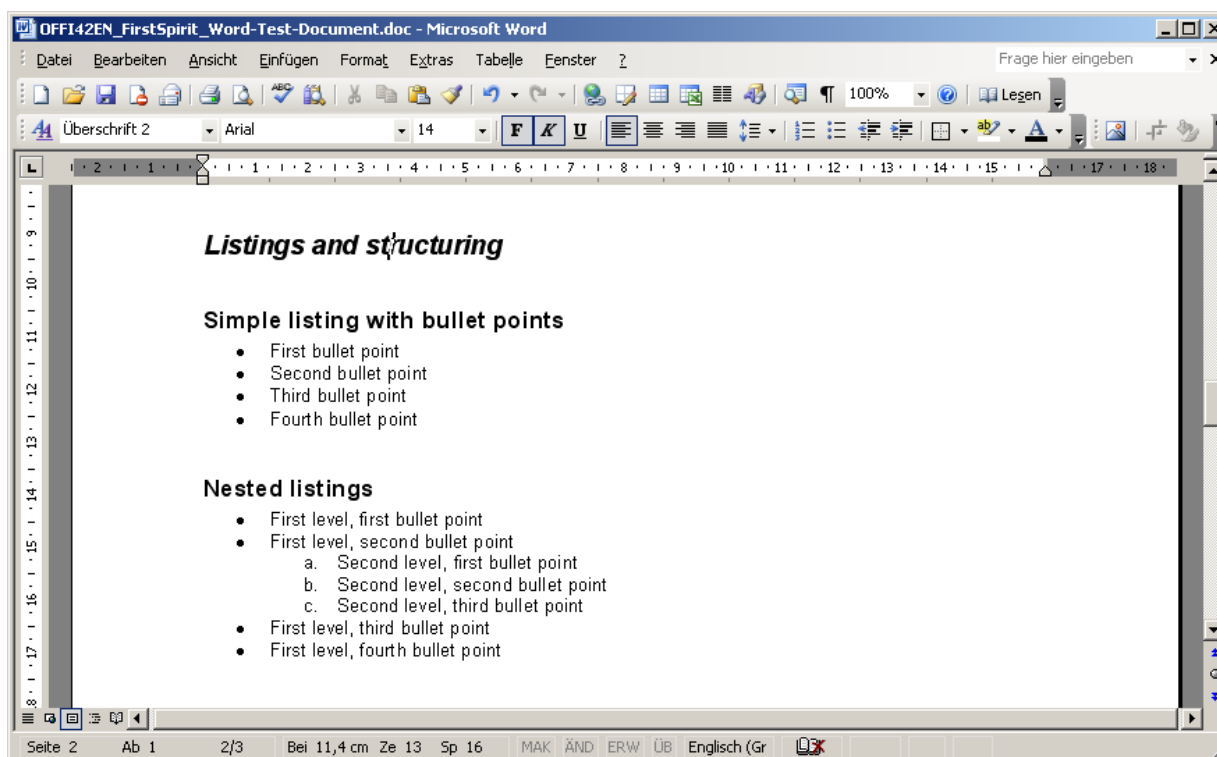


Figure 6-5: Word test document: Lists



6.4.2 Lists in the DOM Editor

After it has been imported using the OfficeConnect function, the text with lists from the test document can be displayed in the DOM Editor as follows:

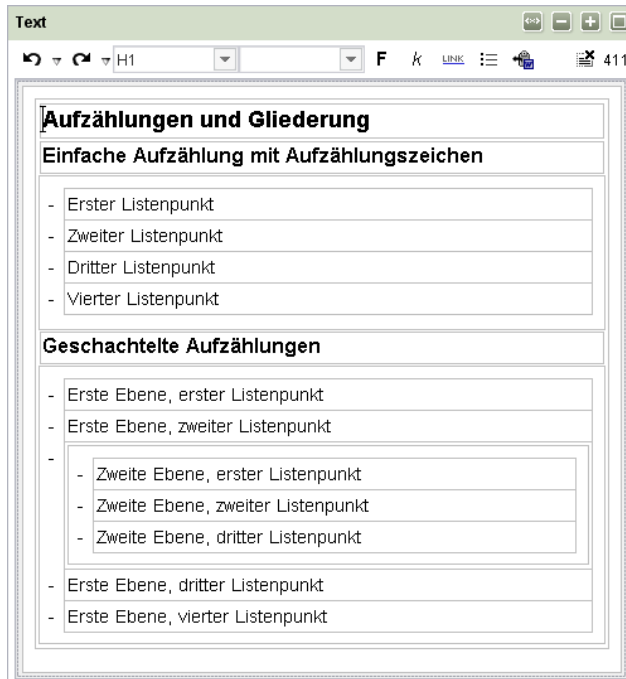


Figure 6-6: DOM Editor with lists from the Word document (DE)



6.5 Importing links

When links are imported they are realised as external links when they are pasted into the DOM Editor:



Figure 6-7: External Link

The display text from the Word document (here "latest news") is imported into the "Link text" field, the link is imported into the "Target URL" field. All fields can be further edited as usual.

6.5.1 Links from Word documents

The following area is marked in the Word test document and is copied onto the clipboard (<CTRL> + <C>):



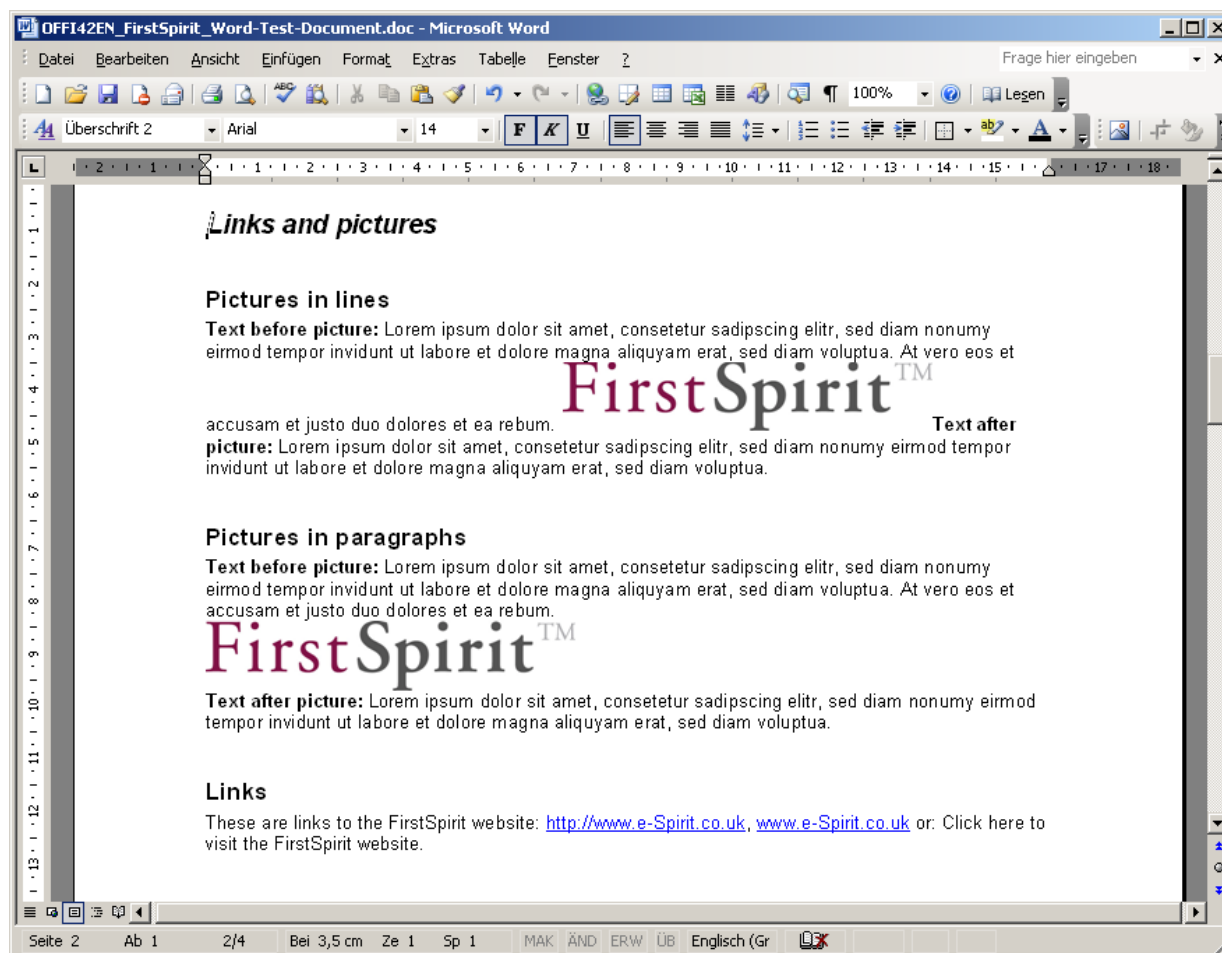


Figure 6-8: Word test document: Links and pictures

6.5.2 Links in the DOM Editor

After it has been imported using the OfficeConnect function, the text with links from the test document can be displayed in the DOM Editor as follows:



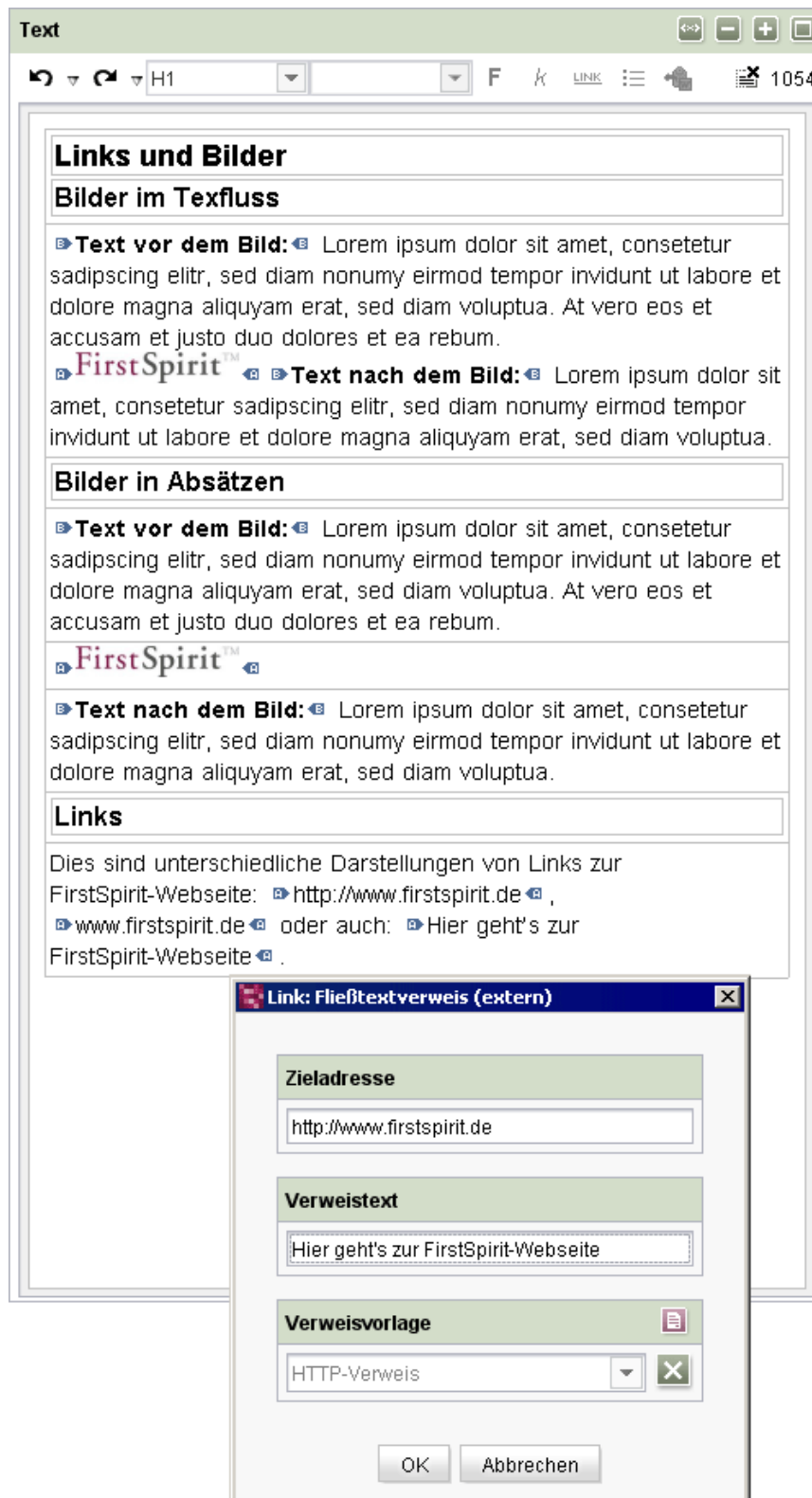


Figure 6-9: DOM Editor with pictures and links from the Word document (DE)



6.6 Importing pictures

6.6.1 Inserting into the DOM editor

If the clipboard contains one or several pictures, they are realised as an internal link when they are pasted into the DOM Editor. A differentiation is made between whether they are embedded in the text body ("pictures in lines") or occupy their own paragraph ("pictures in paragraphs") (see also Figure 6-9):

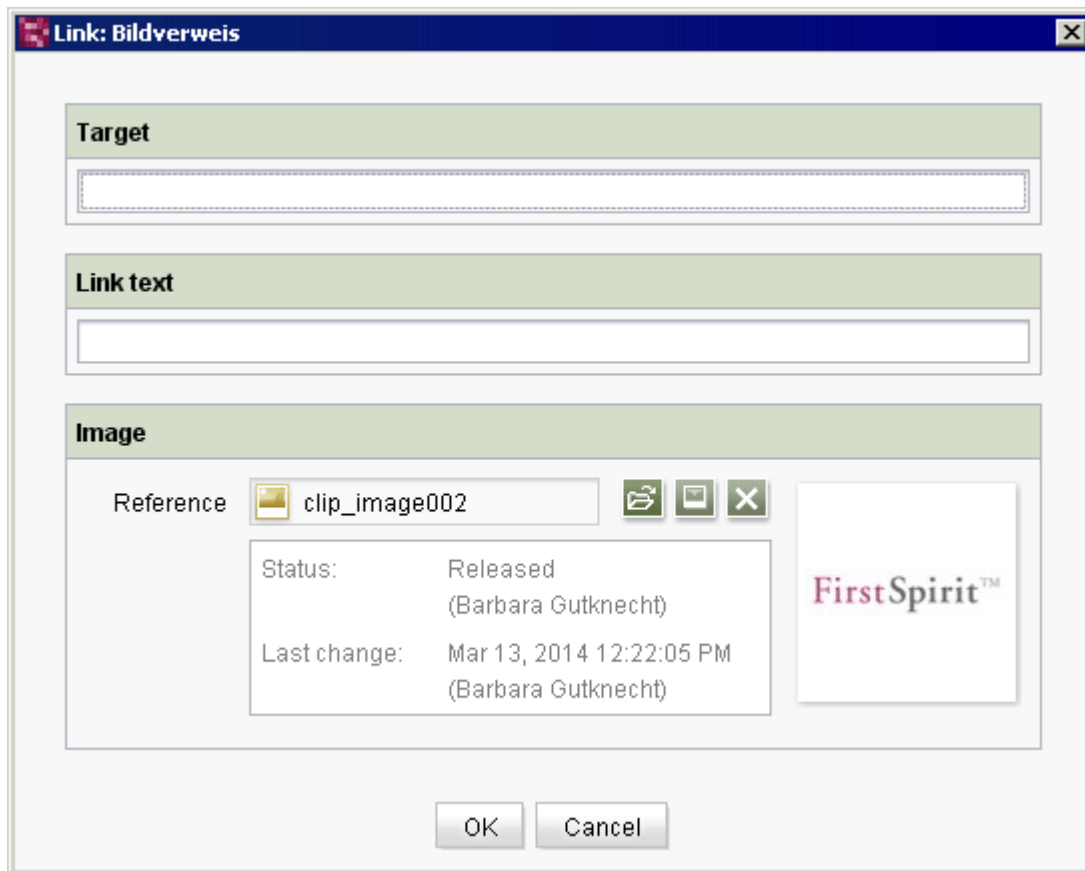


Figure 6-10: Internal link for displaying pictures

This is done using the link template, which is defined in the `mapping` element (see Chapter 4.3.2 page 25). The "Image" field contains the path to the picture in the Media Store (see Chapter 6.6.2 page 80).



6.6.2 Creating pictures in the Media Store

On being imported, the pictures are automatically inserted in the Media Store, namely in sub-folder below the folder "Office import". These folders are created when the pictures are inserted into the DOM Editor.

The UID of the **sub-folder** is formed according to the following schema:

```
import_date_time_user
```

The date is specified in *YYmmDD* format, the time is specified in *HH_MM* format and `User` is the login name of the user who inserted the picture, e.g.

```
import_09040115_16_Admin
```

The inserted **picture** is buffered on the local workstation ("temp" folder). The UID bases on the file name of this temporary picture. It is used without file format (extension), for example:

```
clip_image002
```

This is the same form used to display folders and pictures are they have been added to the tree structure, a display name and a file name for the picture can be individually assigned later. These elements may have to be released in release projects.



Each time pictures are imported from Word documents, they are inserted into the Media Store together with a folder, regardless of whether the same picture already exists or not. If any duplicate pictures are deleted from the Media Store, the reference in the relevant DOM input components in which these pictures are used must be adjusted manually.



6.7 Importing tables

Tables from Word documents can be imported including their formatting. Merges are imported by a ruleset without special configuration.



These should only be inserted into the DOM table or into inline tables (see also documentation about the "FirstSpirit SiteArchitect", Chapter "Tables" and "Integrating tables in the rich text editor").

6.7.1 Tables from Word documents

The following area is marked in the Word test document and is copied onto the clipboard (<CTRL> + <C>):



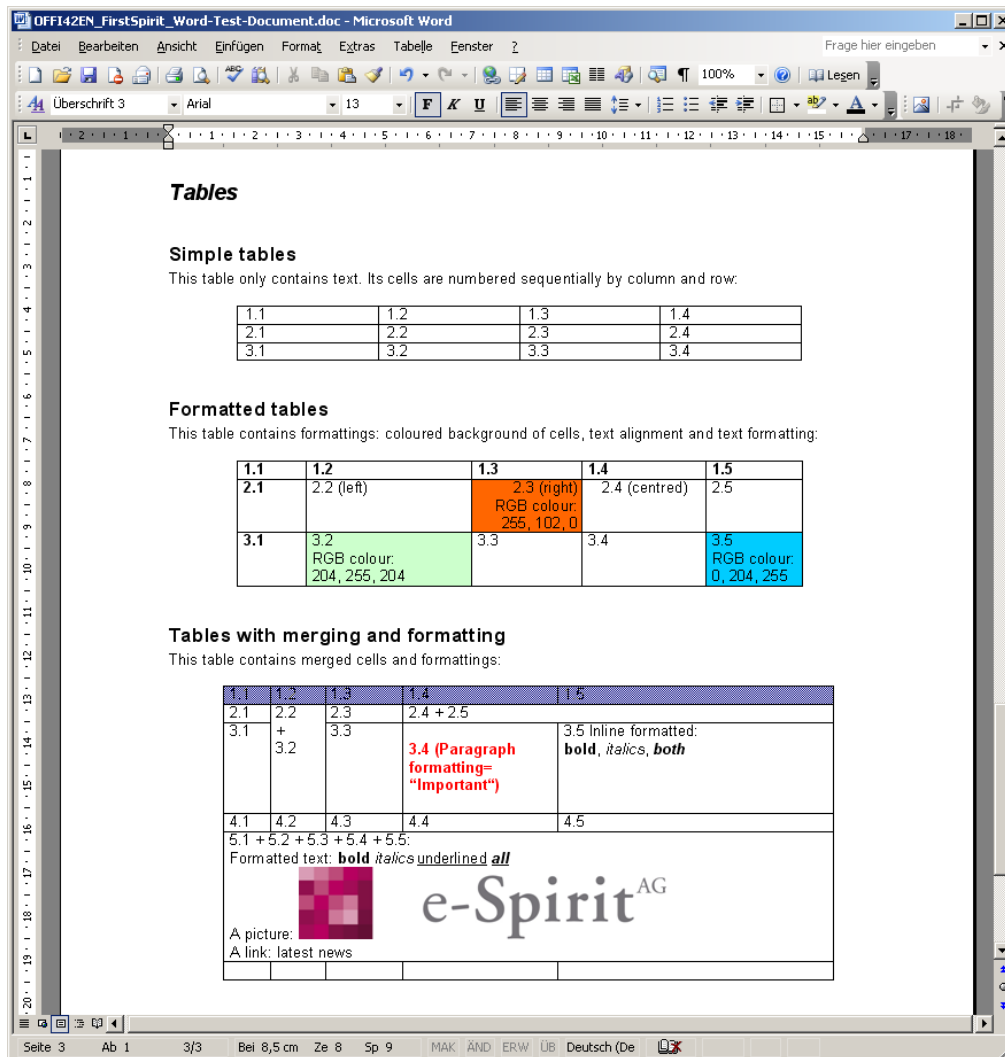


Figure 6-11: Word test document: Tables



6.7.2 Tables in the DOM Editor

After it has been imported using the OfficeConnect function, the text with lists from the test document can be displayed in the DOM Editor as follows:

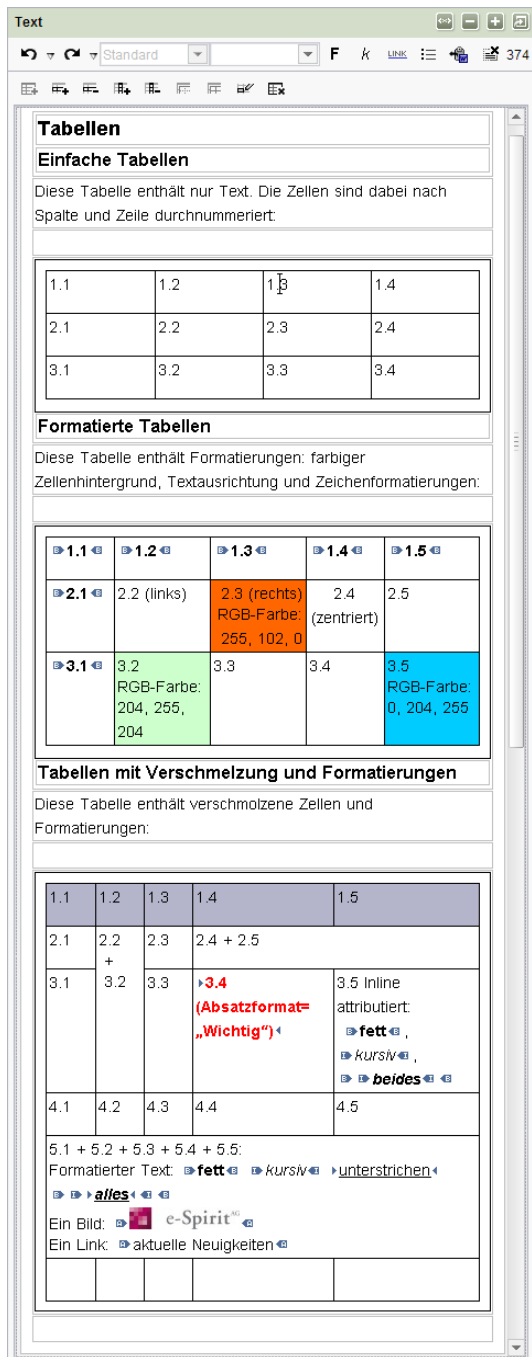


Figure 6-12: DOM Editor with tables from the Word document (DE)



Depending on the specifications of the template developer, the cells in the Word document that do not have specific formatting with regard to background color, font color, and text alignment can also be displayed with other formatting. The cell properties can be edited via the functions available in the editor.

7 Legal notices

The "FirstSpirit OfficeConnect" module is a product of e-Spirit AG, Dortmund, Germany.

The user may only use the module as defined under the terms of the licence agreed with e-Spirit AG.

Details of possible external software products used, not produced by e-Spirit AG, their own licences and any update information, is given on the homepage of each FirstSpirit server in the "Legal Notices" area.

