

# Docgen

(Document Generator)

## User Guide

version 1.9

August 31, 2024

# Important Notice

『Freely available under the terms of the 3-Clause BSD License』

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)

HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

## Document Revision History

Doc Revision Number	Date	Description
1.9	August 31, 2024	add double underline expression
1.8	May 22, 2024	add export file format(html, odt, rtf, xml, txt), and new template installation function
1.7	May 21, 2024	add Excel chart image reference and inline code expression
1.6	May 17, 2024	Markdown can be specified as the input source file
1.5	April 9, 2024	add lua snippet option on command line
1.4	February 5, 2024	add lua function call in paragraph
1.3	November 12, 2021	support excel table
1.2	July 3, 2021	add word's @<property> tag variable
1.1	April 2, 2020	add Visio page to picture & bookmark automation
1.0	February 4, 2019	Initial relase

---

# Table of Contents

<b>1. INTRODUCTION .....</b>	<b>9</b>
1.1 Main functions .....	9
1.2 Document automated creation process .....	10
1.3 How to run .....	10
1.3.1 Project creation example .....	12
1.4 Restrictions on use, licensing and prerequisites .....	12
1.4.1 Limitation .....	12
1.4.2 Licenses .....	12
1.4.3 Prerequisites .....	12
1.5 Known issues .....	13
1.5.1 Localization issue in Excel table .....	13
<b>2. LUA EXPRESSION .....</b>	<b>14</b>
2.1 docgen variables .....	15
2.1.1 Property object .....	15
2.2 AddRevision() function .....	17
2.3 AddTerm() function .....	18
2.4 AddParagraph() function .....	19
<b>3. PARAGRAPH EXPRESSION .....</b>	<b>20</b>
3.1 Title expression .....	20
3.2 Cataloging .....	20
3.3 Text expression .....	21
3.3.1 Bold text .....	21
3.3.2 Italic text .....	22
3.3.3 Underline .....	22
3.3.4 Strikethrough .....	22
3.3.5 Superscript .....	22
3.3.6 Subscript .....	23
3.3.7 Background color .....	23
3.3.8 Text color .....	23
3.3.9 Text size .....	23
3.3.10 Fixed font .....	24
3.3.11 Apply font .....	24
3.3.12 Paragraph style .....	24
3.4 Cross-reference .....	26
3.4.1 Cross-reference in picture or table .....	26
3.4.2 Cross-reference in title .....	26
3.5 Property reference .....	27
3.6 Hyperlink .....	27
3.7 Math equation expression .....	28
3.8 Inline code block .....	28
3.9 Code block .....	29
3.10 Code execution .....	30
3.11 Page separation .....	30
3.12 horizontal line .....	30
3.13 Document activation .....	31
3.14 Lua function call .....	32
3.15 Picture insertion .....	32
3.16 Table insertion .....	34
3.16.1 Table insertion from excel file .....	35

---

3.16.2 Inserting table from Lua expressions .....	36
---	----

---

# List of Tables

Table	Title	Page
Table 2-1.	docgen variables list .....	15
Table 3-1.	Excel table example.....	36
Table 3-2.	Insert only a caption without table content .....	36
Table 3-3.	Lua table example .....	37

---

# List of Figures

Figure	Title	Page
Figure 1-1.	LaTeX setting.....	13
Figure 2-1.	Property tab in WORD .....	16
Figure 2-2.	AddRevision() function example.....	17
Figure 2-3.	AddTerm() function example .....	18
Figure 3-1.	Word's text style modification .....	25
Figure 3-2.	WORD mathematical equation LaTeX conversion .....	28
Figure 3-3.	Donut image example.....	33
Figure 3-4.	SVG vector image example.....	33
Figure 3-5.	Visio vector image example.....	34
Figure 3-6.	Insert only a caption without picture content .....	34
Figure 3-7.	Excel chart example .....	34
Figure 3-8.	"media/table_sample.xlsx" Actual excel file .....	35

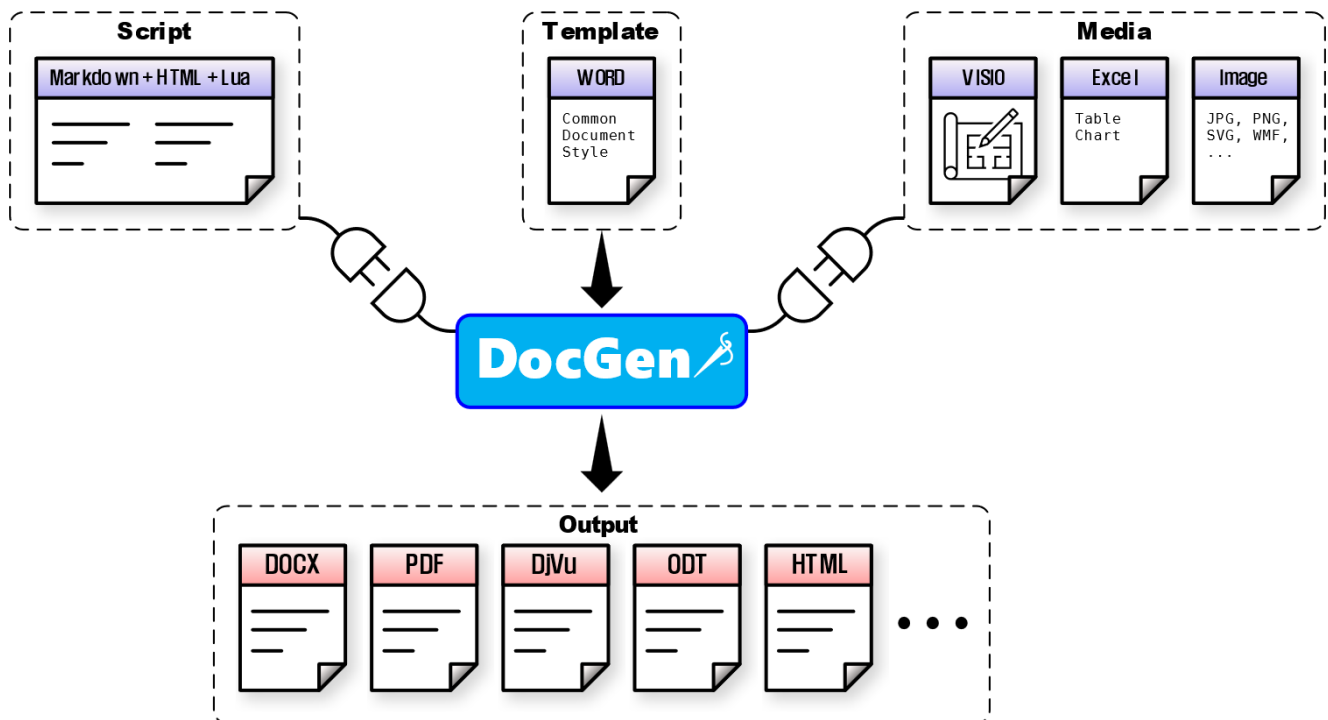
---

## List of Terms

List	Description
TestDrive	TestDrive Profiling Master ( <a href="https://testdrive-profiling-master.github.io/">https://testdrive-profiling-master.github.io/</a> )
Lua	Lua script language ( <a href="#">Wiki</a> , <a href="#">Homepage</a> )
WORD	Microsoft Office's word processor
Excel	Microsoft's spreadsheet editor
Visio	Microsoft Office's Visio - Diagramming and vector graphics application
PDF	Portable Document Format ( <a href="#">Wiki</a> )
markdown	Mark-up language ( <a href="#">Wiki</a> )
VBA	Microsoft's Visual Basic for Application



# 1. Introduction



docgen is a tool that refers to a template WORD (.docx) file and creates a new WORD (.docx) and various documents such as PDF/HTML/RTF through Markdown + HTML + Lua script.

**NOTE:** If you want to skip the introduction process and use it right away, please be sure to refer to '1.4.3. Prerequisites'. If you need new features or find improvements or bugs, please make suggestions to Hyunng-Ki Jeong([clonextop@gmail.com](mailto:clonextop@gmail.com)). Please note that this document was also written and created with docgen.

## 1.1 Main functions

docgen is a tool that automatically creates new WORD documents (.docx) and PDF documents (.pdf) by referring to template word (.docx) files.

Basically, CodeGen allows you to write documents in text format through a grammar compatible with the universal [Lua](#) programming environment and markdown. The main features are listed below.

- Supports various character modulation functions through Lua script
- Supports various document output formats such as WORD, ODT, PDF, and HTML files.
- Automated clause/caption/cross-reference creation function
- Picture, table, style formatting, code block, and mathematical equation expression functions
- Watermark insertion function
- Users can easily change various styles and apply consistent formatting based on Word template documents

### 1.2 Document automated creation process

The document automatedn creation process proceeds in the following steps.

1. Open the template document (template.docx)
2. Apply content from user source (.md or .lua) document
3. Create Word(.docx), PDF(.pdf), DjVu(.djvu), OpenDocument(.odt), HTML(.html), XML(.xml), RichEdit(.rtf), Text(.txt) document

### 1.3 How to run

To run docgen, run as follows.

```
> docgen

Document Generator for TestDrive Profiling Master. v1.8
Usage: docgen [--help] [-t template] [-i template_desc] [-f format] [-l language]
[-r|--run=lua_code] input_file [output_file]

    --help                display this help and exit
    -t template            Document template name/file.
                        *** Installed docgen template list ***
                        testdrive      : TestDrive Profiling Master
                        (default : testdrive)
    -i template_desc       install new docgen template.
                        ex) docgen -t name -i "description" template.docx
    -f format              Extra output format.
                        - supported output format
                        djvu : DjVu image based document
                        html : Standard HTML format
                        odt  : OpenDocument Text format
                        pdf  : Portable Document Format
                        rtf  : Rich Text Format
                        txt  : Plain text format (unicode)
                        xml  : Extensible Markup Language format
                        (ex: -f pdf,html)
    -l language            Document language code string.
                        'docgen.language' variable in Lua
                        (default : 'en')
    -r, --run=lua_code     Run Lua snippet code
    input_file             input Lua or .md(markdown) file
    output_file            output Microsoft Word(.docx) file
```

command line : `docgen INPUT_LUA_FILE OUTPUT_DOCX_FILE`

In `INPUT_LUA_FILE`, set a Lua script or Markdown text file as the input source file. If `OUTPUT_DOCX_FILE` is not specified, it is automatically created appropriately by referring to the given property. The '-t' option specifies the template document that will be the base. If the template document is not specified, the default docgen\_template.docx is specified, but various document types can be created by referencing/changing it.

The '-i' option allows adding a new template document. At this time, 'input\_file' is the new template document source, and 'template' is the target template name to be specified.

The '-f' option specifies saving as a file other than .docx format. You can specify output in combination, such as '-f pdf,html'.

The '-l' option specifies an arbitrary language code. In Lua, you can check it with the 'docgen.language' variable, and in sentences, you can use the expression **'3.13. Document activation'** to select the desired language code.

The '-r' option specifies arbitrary preceding Lua code. This code can contain conditions for changing the document structure.

**NOTE:** The default setting is 'en' (english) and will have no effect on documents that do not use this 'language' setting.

### 1.3.1 Project creation example

아래와 같이 명령어를 입력하여, 간소화된 프로젝트를 생성하고 빌드 할 수 있습니다.

```
> create_project docgen_simplified example
*I: Create DocGen simplified project : 'example'
Run 'build.bat' to build document.

> cd example

> ls
build.bat  main.md  media

> build.bat
1. Introduction
   Main functions
2. first Contents
   media contents
*I: Link all bookmarks.
*I: Build document : Header_Name_userguide_rev1.0.docx
*I: Fields calculation & Saving to PDF output : Header_Name_userguide_rev1.0.pdf
```

## 1.4 Restrictions on use, licensing and prerequisites

### 1.4.1 Limitation

Since WORD's VBA is used, a Windows PC environment with WORD installed is required. (If it is Office 365 for the web, you need to install it as an App. version.) Additionally, in order to directly specify a Visio file (vsd/vsdX) in Picture insertion, Visio installation is required separately. And to use Excel charts, additional installation of Excel is required. Creation is possible on Linux, but functions such as updating entire document fields (can be updated manually by opening directly in WORD)/watermarking/PDF conversion are not performed.

### 1.4.2 Licenses

The source code implemented in docgen complies with the BSD license, and secondary works such as the user's individual scripts or images used to create the document are entirely owned by the user.

### 1.4.3 Prerequisites

Due to limitations in Microsoft Word's setting automation, the math equation expression is initially set to <b>Unicode</b>, the MS standard math equation expression. If you want to express this in "LaTeX", tab 'Equation/Conversions' on the ribbon menu (the menu below must be selected after creating the math equation with 'Insert/Equation' first) as in Figure 1-1. Select 'LaTeX' instead of 'Unicode'. A process is required.

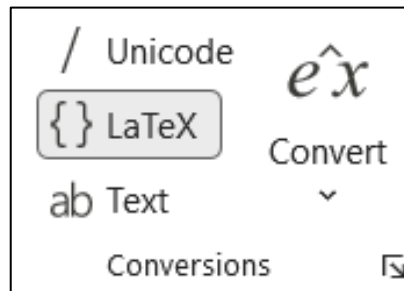


Figure 1-1. LaTeX setting

**NOTE:** Microsoft currently does not provide an automatic change method in this area, and I do not know why the change function is blocked.

## 1.5 Knwon issues

### 1.5.1 Localization issue in Excel table

Regarding the expression of numeric data in Excel, the 'number format' of the default date format originally differs depending on the country setting.

예) For March 8, 2024

```
United state : number format("M/D/YYYY")
              -> '3/8/2024'
Korea        : number format("YYYY-M-D")
              -> 2024-3-8
```

As shown above, the year/month/day display order varies depending on the country setting, but only the US format is used here. If you want to use a different format, you must enter custom 'number format' directly in Excel, not the default format.

# 2. Lua expression

Lua scripts are very lightweight, fast, and simple, making them the preferred script language for non-programmers. There are examples of non-programmer game designers using it in various games such as 'Warcraft WoW' and 'Angry Birds'.

docgen is also written in Lua, and documents can be structured in more detail using Lua syntax in addition to the Markdown+HTML expression.

All functions added in codegen can be used for Lua functions, and the syntax below can be used additionally.

- External Lua-related links
  - [https://en.wikipedia.org/wiki/Lua\\_\(programming\\_language\)/Lua\\_wiki](https://en.wikipedia.org/wiki/Lua_(programming_language)/Lua_wiki)
  - docgen variables
  - AddRevision() function
  - AddTerm() function
  - AddParagraph() function

Here, only the features added to docgen are explained, and documents can be created with minimal template implementation without learning Lua.

### 2.1 docgen variables

Here is a list of variables available in Lua:

**Table 2-1. docgen variables list**

Variable	Type	Description
docgen.property	table	Document property variables table
docgen.contents_only	boolean	Document Title Remove all table of contents and leave only the text (default : false)
docgen.code_format	string	Inline code default format type Refer to. : "3.8. Inline code block" (default : "cpp")
docgen.code_bgcolor	string	Code block background color (24bit hex) (default : F7F7F7)
docgen.code_header_bgcolor	string	Code block header's background color (24bit hex) (default : EDEDED)
docgen.boarder_color	string	(default : F7F7F7)
docgen.fixed_font	string	Default fixed width font font name Refer to. : "3.3.10. Fixed font", "3.9. Code block", "3.8. Inline code block" (default : "Cascadia Mono")
docgen.language	string	Specify the document language Refer to. : "3.13. Document activation" (default : "en")
docgen.table_header.text_color	string	Table header's text color (default : "FFFFFF")
docgen.table_header.bgcolor	string	Table header's background color (default : "808080")
docgen.table_header.height	int	Force to set cell height in table header (default : 0(auto), After applying to the table, it is reset back to 0.)
docgen.table_content.height	int	Force to set cell height in table contents (default : 0(auto), After applying to the table, it is reset back to 0.)
docgen.table_content.scale	float	Force to set table width scale (0 < width_scale <= 1) (default : 1.0, After applying to the table, it is reset back to 1.0.)
docgen.hangul.auto_suffix	boolean	Enable correction of the Korean suffix after bookmark or property - Modify "이/가", "을/를", "은/는" and "와/과" to match the pronunciation. (default : true)

#### 2.1.1 Property object

```
docgen.property["Document_Name"] -- Document name
```

## 2. Lua expression

```
docgen.property["IP_Version"]      -- IP version (ex."1.00")
docgen.property["Main_Title"]      -- Main title name on the first page of
the document
docgen.property["Sub_Title"]        -- Sub title name on the first page of
the document (Can be skipped)
docgen.property["IP_Name_First_Page"] -- Name of the first page of the
document
docgen.property["IP_Name_Header"]   -- Header and name on file
docgen.property["Ownership"]        -- Ownership name
docgen.property["Document_Name_Header"] -- header name (...)
docgen.property["Water_Mark"]       -- watermarking text (Can be skipped)
```

**NOTE:** Sentences starting with "--" is 'comment' notation in Lua. The above property names may vary across documents, the names listed above are examples used in the default TestDrive document and are not fixed names, except for "Water\_Mark".

In Word, there is a 'Property' tab in the menu "File/Information" as like Figure 2-1.

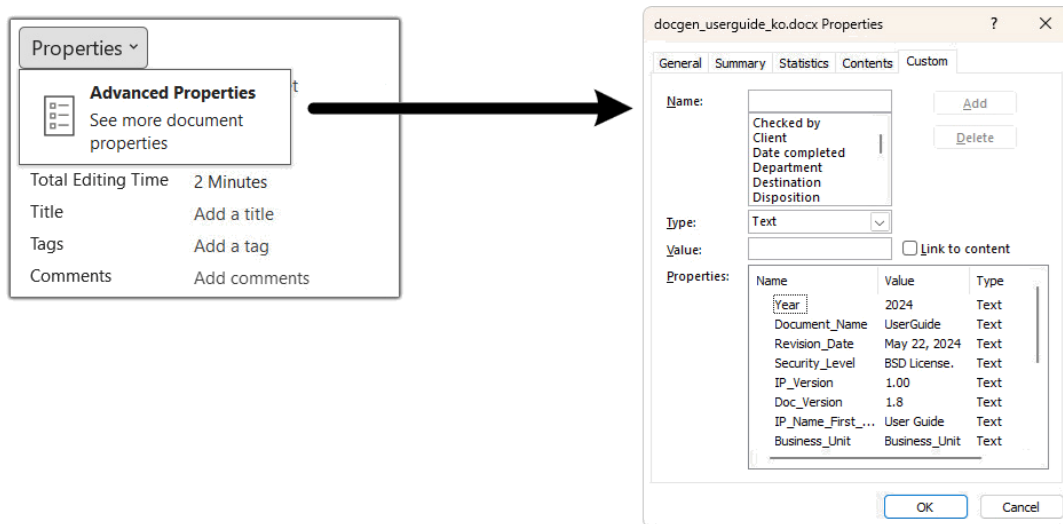


Figure 2-1. Property tab in WORD

In this menu, you can add the **<b>Field</b>** property you want or change the existing **<b>Property</b>**, and the declared **<b>property</b>** is reflected in the **<b>Field</b>** information throughout the document.

Within the lua code, you can change the field value that already exists in the template document in the following way.

Alternatively, direct reference is possible within markdown sentences, such as Property reference.

ex) Example of field designation in this document

```
docgen.property["Document_Name"]      = "UserGuide"
docgen.property["IP_Version"]          = "1.00"
docgen.property["Main_Title"]          = "Document Generator"
docgen.property["IP_Name_First_Page"]  = "User Guide"
docgen.property["IP_Name_Header"]      = "document_generator"
docgen.property["Ownership"]            = "TestDrive"
docgen.property["Document_Name_Header"] = "userguide"
docgen.property["Water_Mark"]          = "TESTDRIVE CONFIDENTIAL"
```



### 2.2 AddRevision() function

**Function prototype : AddRevision(version, year, month, day, description)**

This is a function that manages the version information of a document. They can be listed in version order as shown in the example below, and **description** can use "3. Paragraph expression".

If you never use the AddRevision() function, the "Document Revision History" section is automatically removed.

ex)

```
AddRevision("1.0", 2022, 1, 14, "Initial Draft")
AddRevision("1.1", 2022, 2, 15, "Second release")
AddRevision("1.2", 2022, 3, 16, "Third release")
```

Result)

Document Revision History		
Doc Revision Number	Date	Description
1.2	March 16, 2022	Third release
1.1	February 15, 2022	Second release
1.0	January 14, 2022	Initial Draft

Figure 2-2. AddRevision() function example

### 2.3 AddTerm() function

#### Function prototype : AddTerm(**word**, **description**)

This function inserts a description of a word. It can be used as in the example below, and **Description** can use "**3. Paragraph expression**".

If you never use the AddTerm() function, the "List of Terms" clause is automatically removed.

```
AddTerm("TestDrive", "TestDrive Profiling Master (@<link:https://testdrive-  
profiling-master.github.io/>)")  
AddTerm("Lua", "Lua script language  
(<link:https://en.wikipedia.org/wiki/Lua_(programming_language);Wiki>,  
<link:http://www.lua.org/;Homepage>)")
```

Result)

List of Terms	
List	Description
TestDrive	TestDrive Profiling Master ( <a href="https://testdrive-profiling-master.github.io/">https://testdrive-profiling-master.github.io/</a> )
Lua	Lua script language ( <a href="https://en.wikipedia.org/wiki/Lua_(programming_language)">Wiki</a> , <a href="http://www.lua.org/">Homepage</a> )

Figure 2-3. AddTerm() function example

### 2.4 AddParagraph() function

#### Function prototype : AddParagraph(sentence, [source\_name])

You can fill in the actual document content, and **sentence** can use "3. Paragraph expression".

The contents of **sentence** can be directly described as a string, but the **sentence** can be converted to "[[file\_name]]" refers to an external text file, and direct sentence notation is affected by the POSIX escape character notation, so users unfamiliar with soft programming can use "Example #2)" It is recommended to describe it in an external text file like.

**source\_name** gives the sentence a name. This is used to identify the source when showing real-time status tracking. If the source name is not specified and a file is specified in the sentence, the source name is automatically assigned to the file name.

ex #1) Direct implementation

```
AddParagraph("#Title\  
Paragraph content 1\  
Paragraph content 2", "contents_1")
```

ex #2) External markdown expression text file implementation

```
AddParagraph("[[some.md]]")
```

ex #3) Implemented using the file read function

```
do  
    local txt_contents = String(nil)  
    txt_contents:ReadFile("some.md", false)  
    AddParagraph(txt_contents.s)  
end
```

ex #4) Lua application that checks the 'IsInsert' variable, reads several files,  
changes all "%ABC%" expressions to "good", and inserts sentences.

```
if IsInsert == true then    -- directive check  
    local txt_contents      = String(nil)  
  
    txt_contents:Append(ReadText("some_1.md")) -- some_1.md added  
    txt_contents:Append(ReadText("some_2.md")) -- some_2.md added  
    txt_contents:Append(ReadText("some_3.md")) -- some_3.md added  
    txt_contents:Replace("%ABC%", "good", true) -- change all "%ABC%" to "good"  
  
    AddParagraph(txt_contents.s)    -- Apply to document  
end
```

## 3. Paragraph expression

Paragraph expression refers to how to describe the arguments **description** and **sentence** of the Lua functions of AddRevision, AddTerm, and AddParagraph. The notation method is a mixture of markdown and some html notation, and it additionally has its own grammar for versatility in expression. Additionally, each formula can be used equally wherever sentences are used, such as paragraphs and tables. To allow long sentences to be split into several shorter lines, paragraphs that end with " \" (space + '\') are automatically connected to the next line.

**NOTE:** If you have any suggestions for paragraph expressions or requests for improvement, please contact me at any time. (Refer to 1. Introduction.)

### 3.1 Title expression

**Expression : # Main title**

**## Sub title1**

**### Sub title2**

**#### Sub title3**

**##### Sub title4**

**##### Sub title5**

The document title is displayed by placing the '#' character at the beginning of the line, from **Main title** to **Sub title**. This is the same as markdown. If the '#' character is not at the beginning of the line, it is not recognized as a title and is affected by the "text style" of the WORD's template document. Up to 6 levels can be used.

### 3.2 Cataloging

**Expression : \* Element1**

**\*\* Element2**

**\*\*\* Element3**

**\*\*\*\* Element4**

**\*\*\*\*\* Element5**

**\*\*\*\*\* Element6**

Cataloging is similar to markdown, but there are some differences for more expressive functions. If the '\*' character exists at the beginning of a line, it is displayed as a list, and the indentation is determined by the number of '\*'s.

You can also use '>' after the '\*' character listed for a number or special expression to remove the existing marker

### 3. Paragraph expression

---

and use a different form of expression. Up to 6 levels can be used.

ex)

```
* first element
** second element
*** third element
**** fourth #1 element
**** fourth #2 element
**** >fourth #2 element extension
**** @<b>@<color:FF0000>Cataloging@</color>(目錄, List)@</b> refers to writing down
names of people, names of products, book titles, tables of contents, and items to be
checked in a certain standard and order to make them easier to read.
*> 1) first number element
**> 1-1) second number element
*> 2) first number element
**> 2-1) second number element
***> Unmarked Element Extended element
```

Result)

- first element
  - second element
    - ◆ third element
      - fourth #1 element
      - fourth #2 element
      - fourth #2 element extension
      - **Cataloging**(目錄, List) refers to writing down names of people, names of products, book titles, tables of contents, and items to be checked in a certain standard and order to make them easier to read.
- 1) first number element
  - 1-1) second number element
- 2) first number element
  - 2-1) second number element
  - Unmarked Element Extended element

### 3.3 Text expression

Text expressions can specify the color, thickness, italics, underline, size, superscript/subscript, etc. of the text. The scope of application is limited to one line, and multiple formula notations may be expressed overlapping.

#### 3.3.1 Bold text

**Expression : @<b>expression@</b>**

Bold text is expressed by surrounding **Expression** with a 'b' tag, similar to HTML.

ex)

```
@<b>It's in bold.@</b> It's not in bold.
```

Result : **It's in bold.** It's not in bold.

#### 3.3.2 Italic text

**Expression :** @<i>expression@</i>

Italic text is expressed by surrounding the **expression** with an 'i' tag, similar to HTML.

ex)

```
@<i>italic@</i> Non italic
```

Result : *italic* Non italic

#### 3.3.3 Underline

**Expression(single) :** @<u>expression@</u>

**Expression(double) :** @<U>expression@</U>

The underlined text is expressed by surrounding **expression** with a 'u' tag, similar to HTML. When you write a formula in capital letters, it is double underlined.

ex)

```
@<u>underlined text@</u>, @<U>double underlined text@</U>
```

Result : underlined text, double underlined text

#### 3.3.4 Strikethrough

**Expression :** @<s>expression@</s>

The strikethrough text is expressed by surrounding **expression** with 's' tags, similar to HTML.

ex)

```
@<s>Strikethrough text@</s>
```

Result : ~~Strikethrough text~~

#### 3.3.5 Superscript

**Expression :** @<sup>expression@</sup>

Superscript text is expressed by surrounding **expression** with 'sup' tags, similar to HTML.

ex)

```
Text@<sup>Superscript@</sup>
```

Result : Text<sup>Superscript</sup>

#### 3.3.6 Subscript

**Expression :** @<sub>expression@</sub>

Subscript text is expressed by surrounding **expression** with 'sub' tags, similar to HTML.

ex)

```
Text@<sub>Subscript@</sub>
```

Result : TextSubscript

#### 3.3.7 Background color

**Expression :** @<bgcolor:color\_value>expression@</bgcolor>

To change the background color, the expression is **expression** surrounded by 'bgcolor' tags similar to HTML, and the color is specified by **color\_value** Expressed in 24bit RGB hexadecimal.

ex)

```
@<bgcolor:FF0000>Red background@</bgcolor> expression
```

Result : Red background expression

#### 3.3.8 Text color

**Expression :** @<color:color\_value>expression@</color>

To change the font color, the expression is **expression** surrounded by 'color' tags similar to HTML, and the color is specified by **color\_value** Expressed in 24bit RGB hexadecimal.

ex)

```
@<color:FF0000>Red text@</color> expression
```

Result : Red text expression

#### 3.3.9 Text size

**Expression :** @<size;font\_size>expression@</size>

To change the font size, the expression is **expression** surrounded by 'size' tags similar to HTML, and the color is specified by **font\_size** Specify it in point units.

ex)

```
@<size:30>Big text@</size> @<size:10>small text@</size>
```

Result : Big text small text

### 3. Paragraph expression

---

#### 3.3.10 Fixed font

**Expression :** @<fixed>**expression**@</fixed>

To use fixed font, the **expression** is surrounded by 'fixed' tags similar to HTML.

**NOTE:** Fixed-width fonts can be specified with Lua's '**docgen.fixed\_font**' variable, and the default font name is 'Cascadia Mono'.

ex)

```
@<fixed>Fixed font@</fixed> expression.  
Result : Fixed font expression.
```

#### 3.3.11 Apply font

**Expression :** @<font:**font\_name**>**expression**@</font>

You can express special characters by specifying the font of "font\_name". To apply specific font, the **expression** is surrounded by 'font' tags similar to HTML.

ex)

```
A @<font:Wingdings>à@</font> B  
결과 : A → B
```

#### 3.3.12 Paragraph style

**Expression :** :::**style\_name,alignment**

In Microsoft Word, you can add/edit your own style formats as shown below. In docgen, you can apply the style formats that exist in the template document to paragraph units. In other words, it means that users can create/apply various style formats they want.



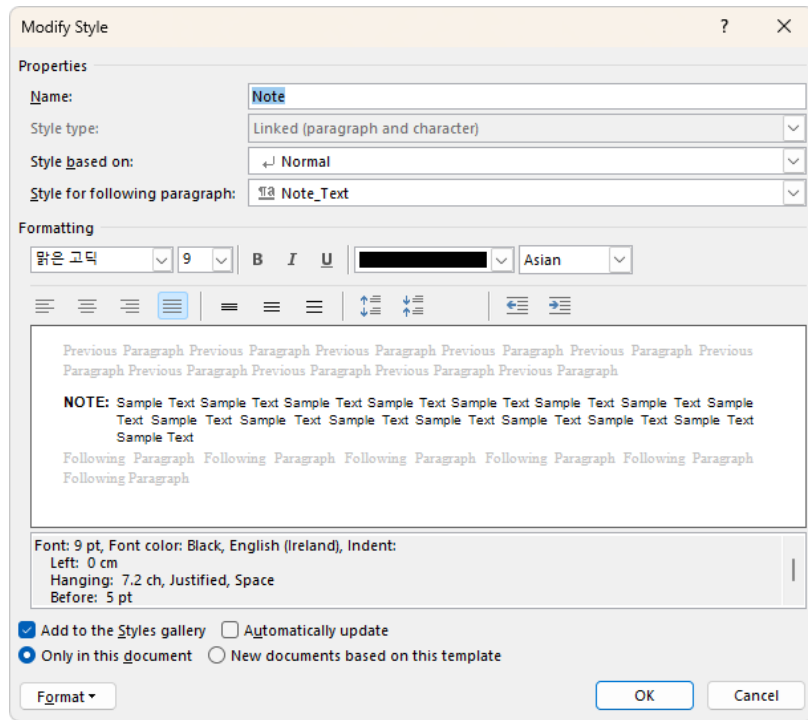


Figure 3-1. Word's text style modification

Change the paragraph style of the next line to '**style\_name**'. The style changes for only one line, and reverts to the default style formatting for the next line. Style formatting is implemented by referencing the style specified in the initial template WORD document, and can be changed or added by searching for 'text style' in WORD. Style names include tab characters, "/", "--", or ";". Anything after the character is ignored. It doesn't matter if you enter the style ID rather than the style name.

'**alignment**' can be one of 'left', 'right', 'center', and If omitted, basically it is considered as two-sided alignment. And you can also specify only '**alignment**' without 'style\_name'.

ex) Apply note style

```
 :::Note          -- Additional explanation
Including both civilian and military deaths, an estimated 60 to 70 million \
people died as a result of World War II. \
In the aftermath of this war, collectivistic ideology, which had been \
the mainstream of society, declined in the Western world and individualistic \
ideology emerged, which continues to this day.

 :::Warning
Keep warning! Keep warning! Keep warning! Keep warning! Keep warning! Keep warning!
Keep warning! Keep warning! Keep warning! Keep warning!

 :::Caution
Keep caution! Keep caution! Keep caution! Keep caution! Keep caution! Keep caution!
Keep caution! Keep caution! Keep caution! Keep caution!

 :::center
center alignment

 :::right
right alignment
```

### 3. Paragraph expression

---

Result)

**NOTE:** Including both civilian and military deaths, an estimated 60 to 70 million people died as a result of World War II. In the aftermath of this war, collectivistic ideology, which had been the mainstream of society, declined in the Western world and individualistic ideology emerged, which continues to this day.

---

**Warning:** Keep warning! Keep warning! Keep warning! Keep warning! Keep warning! Keep warning! Keep warning! Keep warning! Keep warning! Keep warning! Keep warning!

---

---

**Caution:** Keep caution! Keep caution! Keep caution! Keep caution! Keep caution! Keep caution! Keep caution! Keep caution! Keep caution! Keep caution! Keep caution!

---

center alignment

right alignment

### 3.4 Cross-reference

**Expression :** @<bookmark:target>

Implement cross-referencing of titles, pictures, tables, etc.

Just write the caption or title of the figure or table in target.

#### 3.4.1 Cross-reference in picture or table

ex)

```
@<bookmark:LaTeX setting>
```

Result : Figure 1-1

#### 3.4.2 Cross-reference in title

In the case of titles, cross-references can be expressed in four ways depending on the presence or absence of a prefix, as shown below.

1. Reference to general sentences (expressed without a separate prefix)  
: target
2. Page number reference  
: &target
3. Chapter/section number reference  
: #target
4. Chapter/section number and full sentence reference

### 3. Paragraph expression

---

: **@target**

The example below shows an actual usage example, and you can see that clicking on the sentence moves to the link to the current document.

ex) Reference to general sentences

```
@<bookmark:Restrictions on use, licensing and prerequisites>
```

Result : Restrictions on use, licensing and prerequisites

ex) Page number reference

```
@<bookmark:&Restrictions on use, licensing and prerequisites>
```

Result : 12

ex) Chapter/section number reference

```
@<bookmark:#Restrictions on use, licensing and prerequisites>
```

Result : 1.4

ex) Chapter/section number and full sentence reference

```
@<bookmark:@Restrictions on use, licensing and prerequisites>
```

Result : 1.4. Restrictions on use, licensing and prerequisites

### 3.5 Property reference

**Expression : @<property:property\_name>**

You can create properties with various names using 'Property object' among 'Lua expression', and it provides a way to quote these in the text.

예)

```
@<property:Main_Title>, @<property:Ownership>
```

Result : Docgen, clonextop@gmail.com

### 3.6 Hyperlink

**Expression : @<link:URL\_target;display\_text>**

Hyperlinks are displayed as above, and **display\_text** can be skipped.

ex) "https://testdrive-profiling-master.github.io/" hyperlink expression

```
@<link:https://testdrive-profiling-master.github.io/;TestDrive Profiling Master>
```

Result : [TestDrive Profiling Master](https://testdrive-profiling-master.github.io/)

#### 3.7 Math equation expression

##### Expression : `$$math_formula$$`

In the case of mathematical formulas, the mathematical equation is displayed by surrounding it with "\$\$", as in markdown.

But, WORD's mathematical equation is initially set to the expression "**Unicode**". Therefore, if you want to use the equation "**LaTeX**", please note that you must change the settings by referring to "**1.4.3. Prerequisites**".

When a mathematical equation is expressed as a single expression, it is displayed centered, but when expressed in the middle of a sentence, the expression is automatically changed to match the sentence, and the example below is implemented in LaTeX expression, so if the result is notated incorrectly, it is the result of not attempting the "Prerequisites" above.

ex) 2 types of mathematical equation expressions

```


$$f(x) = a_0 + \sum_{n=1}^{\infty} \left( a_n \cos\left\{\frac{n\pi}{x}\right\} + b_n \sin\left\{\frac{n\pi}{x}\right\} \right)$$


```

The expression in the sentence changes to

```

 $f(x) = a_0 + \sum_{n=1}^{\infty} \left( a_n \cos\left\{\frac{n\pi}{x}\right\} + b_n \sin\left\{\frac{n\pi}{x}\right\} \right)$ 

```

Result :

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left( a_n \cos\left\{\frac{n\pi}{x}\right\} + b_n \sin\left\{\frac{n\pi}{x}\right\} \right)$$

The expression in the sentence changes to  $f(x) = a_0 + \sum_{n=1}^{\infty} \left( a_n \cos\left\{\frac{n\pi}{x}\right\} + b_n \sin\left\{\frac{n\pi}{x}\right\} \right)$ .

WORD's mathematical equation can be changed by first creating the equation as shown in Figure 3-2 and then pressing the 'Linear format' button in the menu. You can write this string as a mathematical equation by adding the characters "\$\$" to both ends.

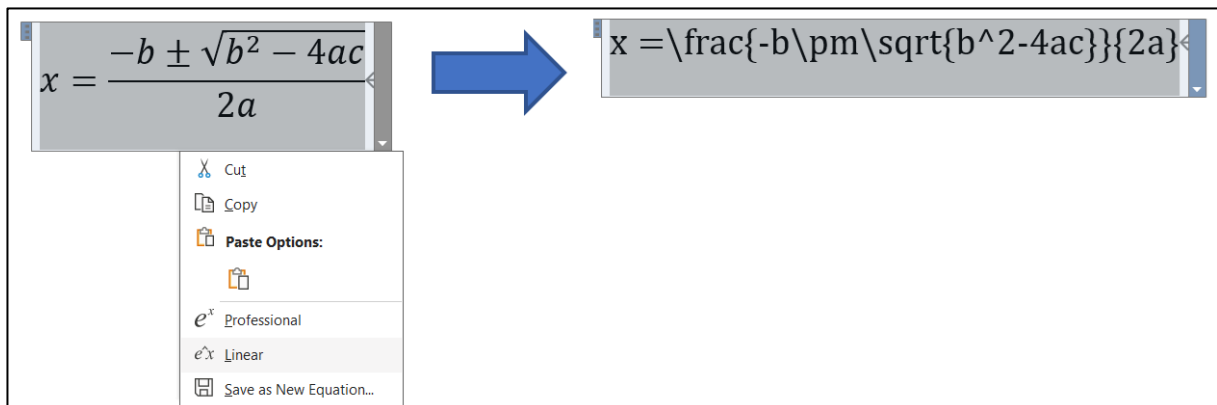


Figure 3-2. WORD mathematical equation LaTeX conversion

#### 3.8 Inline code block

##### Expression : `@<code:code_format>inline_code@</code>`

### 3. Paragraph expression

Statement Represents inline code. '`code_format`' can be omitted and, the default code format is 'cpp', and you can use the same code formats supported by 'Code block'.

**NOTE:** You can change the default code format in Lua by specifying '`docgen.code_format`'. Initial value of code format is 'cpp'.

ex)

Examples of output from C expressions is "`@<code>printf("Hello world.");@</code>`".

Result :

Examples of output from C expressions is "`printf("Hello world.");`".

### 3.9 Code block

**Expression :** ```code_format:header`

`code_contents ...`

````

In the case of `code_format`, you can check the supported code formats by entering the "`code_highlighter -l`" command as shown below. `header` is used to insert text above the `code_contents` and can be omitted.

```
> code_highlighter -l
*** Available current input code format(C:/Project/Profiles/Common/bin/source-highlight-lang/lang.map) list

C, F77, F90, H, ac, ada, adb, am, applescript, asm
autoconf, awk, bash, bat, batch, bib, bison, c, caml, cbl
cc, changelog, clipper, cls, cobol, coffee, coffeescript, conf, cpp, cs
csh, csharp, css, ctp, cxx, d, desktop, diff, dmd, docbook
dtx, el, eps, erl, erlang, errors, f, f77, f90, feature
fixed-fortran, flex, fortran, free-fortran, glsl, go, groovy, h, haskell,
haxe
hh, hpp, hs, htm, html, hx, hxx, in, ini, ipxe
islisp, java, javalog, javascript, js, json, kcfg, kdevelop, kidl, ksh
l, lang, langdef, latex, ldap, ldif, lex, lgt, lhs, lilypond
lisp, ll, log, logtalk, lsm, lua, ly, m4, makefile, manifest
mf, ml, mli, moc, opa, outlang, oz, pas, pascal, patch
pc, perl, php, php3, php4, php5, pkgconfig, pl, pm, po
postscript, pot, prg, prolog, properties, proto, protobuf, ps, py, python
r, rb, rc, rs, ruby, s, scala, scheme, scm, scpt
sh, shell, sig, sl, slang, slsh, sml, spec, sql, sty
style, syslog, systemverilog, tcl, tcsh, tex, texi, texinfo, tk, tml
txt, ui, upc, v, vala, vbs, vbscript, verilog, vim, xhtml
xml, xorg, y, yacc, yaml, yml, yy, zsh
```

Below is an example of quoting 'C' code. To use the string "" when quoting, you can quote it by writing "@``". To insert a line number in the code, you can add '#' and quote it like this: "`#code_format`".

ex) 'C' Code block with line numbers

```
``#C : @<color:FF0000>[</color>main.cpp<color:FF0000>]</color>
```

### 3. Paragraph expression

---

```
#include <stdio.h>

int main(int argc, const char* argv[]) {
    printf("Hello World.\n");
    return 0;
}
...
```

Result)

[main.cpp]

```
1: #include <stdio.h>
2:
3: int main(int argc, const char* argv[]) {
4:     printf("Hello World.\n");
5:     return 0;
6: }
```

#### 3.10 Code execution

**Expression :** ````[code_format]`  
`code_contents`  
`````

It has a similar format to 'Code block', but if the "code format" is surrounded by "[ ]", the code content is executed.

The currently available code format supports only "Lua".

ex)

```
```[lua]
property["Document_Name"] = "UserGuide"
property["IP_Version"] = "1.00"
```
```

#### 3.11 Page separation

**Expression :** `;;;`

If you enter at least three ';' characters at the beginning of a line, it will advance to the next page.

#### 3.12 horizontal line

**Expression :** `---`

If you enter at least three '-' characters at the beginning of a line, a horizontal line will be entered, just like in markdown.

### 3. Paragraph expression

---

ex)

---

Result)

---

#### 3.13 Document activation

**Expression #1 : %%% language\_code**

**Expression #2 : %%% (Lua\_code)**

Document activation determines whether the following statements will be applied to the actual document. The condition for document activation can be specified as a return value of "language\_code name" or "Lua\_code" enclosed in parentheses "()".

In the case of "language\_code", it is compared with the 'language code name' specified in the '-l' option. (Refer to '1.3. How to run'). If not equal, the next document content is ignored until the next document activation specified. This ensures that only sentences written in your desired language can be used in your document.

For "Lua\_code", it must be enclosed in parentheses "()". If the return value of Lua code execution is 'true'(boolean), then use the following document contents. If not same as 'true', the next document contents will be ignored.

If it ends with '%%%' without language code or Lua code definition, the document content is activated from the next line.

**NOTE:** Language code names are case-sensitive, and you can improve readability by listing '-', '=', or '%' characters after the language code name. The default value is 'en'. The language code can be used as the 'docgen.Language' variable in Lua.

ex1)

```
%%% ko -----
한글 출력입니다.
%%% en -----
It's an english output.
%%% -----
It's all output.
```

Result) When using the '-l en' option.

```
It's an english output.
It's all output.
```

ex2)

[Lua code]

```
test_mode = true
```

[Markdown code]

```
%%% (test_mode == true)
test_mode is activated.
```

### 3. Paragraph expression

---

```
%% (test_mode ~= false)
test_mode is inactivated.
%%
```

Result)

```
test_mode is activated.
```

#### 3.14 Lua function call

##### Expression : @<lua:lua\_function>

Call a Lua function. If the returned value is in string type, it is applied to the next document content.

ex)

```
Lua variable 'docgen.language' is @<lua:docgen.language>
```

Result)

Lua variable 'docgen.language' is en

#### 3.15 Picture insertion

##### Expression : @<img:filename;scale;caption>

Pictures support jpg, png, bmp, gif, tif, svg, wmf, and vsd/vsdX (Visio installation required), xls/xlsx(Excel installation required) formats. **file name** specifies the file name to display, if the picture requires an outline border, indicate it as "**#filename**". In the case of a Visio file, a separate page name can be specified as "**filename[page name]**", and if **page name** is not specified, it is considered the first page.

In the case of an Excel file, By specifying "**file\_name[sheet\_name[:chart\_title]]**", you can designate the chart of a specific sheet as an image. (The chart title can be omitted. If omitted, the first chart on the sheet is used.)

Additionally, in the case of **scale**, the maximum value based on the width of the paper is 1.0, and you can specify the size by entering a value between 0 and 1. If not specified, the default 1.0 is assumed.

**caption** literally specifies the caption content and can be used in '3.4. Cross-reference'. Captions may be omitted.

ex1) Illustration of a donut with border in size  $\frac{1}{4}$

```
@<img:#media/donut.jpg;0.25;Donut image example>
```

Result)





**Figure 3-3. Donut image example**

ex2) Borderless vector image

```
@<img:media/awesome_svg.svg;0.7;SVG vector image example>
```

Result)



**Figure 3-4. SVG vector image example**

ex3) Insert Visio specific page as vector image

```
@<img:media/test.vsd[test_sample];0.8;Visio vector image example>
```

Result)

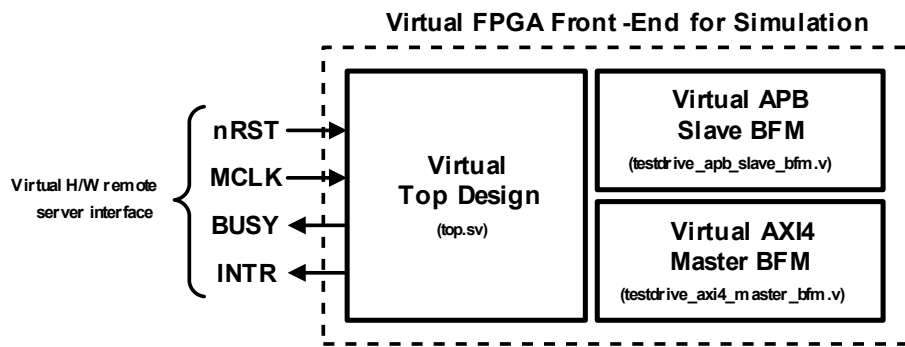


Figure 3-5. Visio vector image example

ex4) Insert only a caption without a picture

```
@<img:nil;0;Insert only a caption without picture content>
```

Result)

Figure 3-6. Insert only a caption without picture content

ex5) Insert chart image from Excel

```
@<img:media/chart_sample.xlsx[chart_population:Population in the world];0.7;Excel chart example>
```

Result)

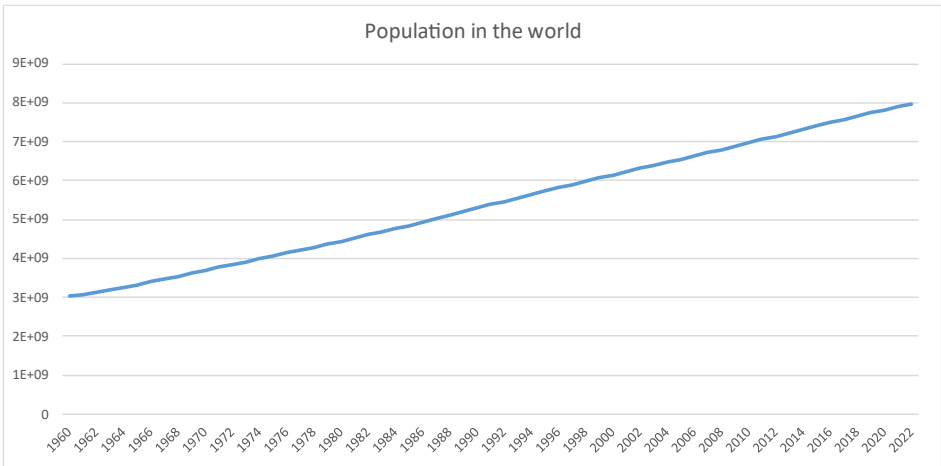


Figure 3-7. Excel chart example

3.16 Table insertion

Tables support two expression methods: Excel and Lua script.

### 3. Paragraph expression

If the table is too long and you move to the next page, the header is automatically repeated.

#### 3.16.1 Table insertion from excel file

**Expression :** `@<tbl:file_name;sheet_name;caption>`

In the case of inserting a table, you can move the content created in Excel, and the width will be the same as possible for the cells merged in Excel. The first line of the table is the top title, and when you want to align lower data columns in the center, insert the character '\*' front of the title name. Captions may be omitted.

By default, the width of the table is displayed so that it always appears full horizontally, but by modifying the `docgen.table_content.scale` variable of `docgen` variables, you can arbitrarily reduce the horizontal width.

**NOTE:** Figures or tables are basically grouped together with captions as one object so that they are not displayed separately from the caption. Therefore, when they cannot be displayed at once in the remaining page area, they are displayed on the next page. However, for very long tables, the extra blank space at the beginning can look awkward. In this case, you can prevent this by inserting the '#' character in front of the 'file\_name'.

ex1) Table from excel file : "media/table\_sample.xlsx"

```
@<tbl:media/table_sample.xlsx;Sheet1;Excel table example>
```

|   | A     | B            | C  | D |
|---|-------|--------------|--|---|
| 1 | Index | Name         | Description  |   |
| 2 | 0     | <u>Ace</u>   | description #0   |   |
| 3 | 1     | <i>Taro</i>  | <b>description #1 bold colored <i>italic underline</i></b> |   |
| 4 |       | <b>Bison</b> | description #2   |   |
| 5 | 2     | <i>Toss</i>  | @<b>description @<color:FF0000>#3@</color>@</b>            |   |
| 6 | 3     | Gran         | description #4   |   |
| 7 | 4     | Mobi         | description #5   |   |
| 8 | 5     | Dose         | description #6   |   |
| 9 |       |              |  |   |

Figure 3-8. "media/table\_sample.xlsx" Actual excel file

Result)

### 3. Paragraph expression

Table 3-1. Excel table example

| Index | Name        | Description   |
|-------|-------------|---|
| 0     | <u>Ace</u>  | description #0  |
| 1     | Taro        | <b>description #1 bold colored italic underline x<sup>y</sup></b> |
| 2     | Bison       | description #2  |
|       | Toss        | <b>description #3</b>   |
| 3     | Gran        | description #4  |
| 4     | Mobi        | description #5  |
| 5     | <u>Dose</u> | description #6  |

ex2) Insert only captions without table content

```
@<tbl:nil;nil;Insert only a caption without table content>
```

Result)

Table 3-2. Insert only a caption without table content

#### 3.16.2 Inserting table from Lua expressions

**Expression : @<tbl:lua;table\_variable\_name;caption>**

Basically, since docgen is written based on Lua scripts, you can create a table with Lua variables and utilize it.

This is an example that utilizes the lua table already defined as shown below.

예) declared lua expression

```
lua_table_example = {
  HeaderCount = 2,
  {"Head A", merge={1,2}}, {"Head B", merge={3,1}}, {width=300,
style="TableTextCenter"}, "",
  {"", "Sub A", "Sub B", "Sub C"},
  {"index1", "a", "1", "aa"},
  {"index2", "b", "2", "bb"},
  {"index3", {"c", merge={2,1}}, "", "@<color:ff0000>cc@</color>"}
}
```

예) Expression within paragraph

```
@<tbl:lua;lua_table_example;Lua table example>
```

결과)

**Table 3-3. Lua table example**

| Head A | Head B |       |       |
|--------|--------|-------|-------|
|        | Sub A  | Sub B | Sub C |
| index1 | a      | 1     | aa    |
| index2 | b      | 2     | bb    |
| index3 | c      |       | cc    |

Lua tables are expressed with multiple braces, such as "{ { { ... }, ... }, ... }". The outer curly braces indicate the table, the next brace indicates Row, and the next brace indicates Column.

When expressing Column and Row, you can specify the following properties. In the case of Column, if you describe it only as a string, the braces can be omitted.

- Lua table Row properties
  - HeaderCount: Specifies the number of header lines. (Default: 1)
- Lua table Column properties
  - data : Specifies the string data of the cell. It is possible to omit "data=" and describe only as a string. However, if omitted, it must be the first attribute of the cell technology. (default : " ")
  - width : Specifies the horizontal length of one column. The final relative length is given by dividing it from the total length of all columns. This property is valid only for the Column described in the first Row. (Default: 100)
  - style : Specifies the text style name to apply to the column. The specified style is equally applied to the same child columns. (Default: "TableTextLeft", when specifying the style back to default, specify "" rather than nil.)
  - merge : It functions to merge multiple cells within a certain range starting from the current cell location. Range can be specified in properties as {width,height}.