TEMPLATE$

TEMPLATE$ returns a string value containing formatted values and lines of text

Syntax

TEMPLATE$(Template$, ReplaceMode%=0,1,2, [InpVar])

Arguments

- Template$ is a string/document containing the textual skeleton of the document to be created, including parameters and format statements
- Replace Mode% is an option how to replace the parameters in the template when "~" is used:
  - Replacemode% = 0: first remove *"~ParName"* and then **overwrite** the value of *ParName* over the remaining text starting on position of ~
  - Replacemode% = 1: first overwrite *"~ParName"* with spaces (the amount specified as Cell width in the Slots & Properties of the parameter) and then **overwrite** the value of*ParName* over the remaining text starting on position of ~
  - Replacemode% = 2: first remove *"~ParName"* and then **insert** the value of *ParName*
- InpVar is an optional list of parameters present as labels in Template$ to be replaced by their values.

Remarks

1. The TEMPLATE$ function is the core function to create knowledge based scripts for other programs;
2. The basic mechanism of the TEMPLATE$ function is a script including parameter names, which will be replaced with actual values during execution;
3. Please note that, when no format tags are used (see hereafter), the Cell width specified in the the Slots & Properties will have influence on the placement of the parameter values;
4. Furthermore, string values are aligned to the left and numeric values are aligned to the right by default;
5. Although you can write the script directly between quote on the place of *Template$*, the easiest way is to use the TEXTITEM$() function to refer to a script in the data slot of the relation;
6. Please make sure that, when you are using TEXTITEM$() you are using delimiters for the start and finish of the script that are different from other characters in your template!
7. There are several ways to define where and how to place a parameter value:
8. a. ~ParameterName, see also syntax;
   b. ~ObjectName(ParameterName1, , ~ParameterName or ~TeLiTabName(ParameterName1, , ~ParameterName, write the content of an object/TeLiTab (most probably a table) specified by the parameter names in their defined order. Useful for writing tables;
   c. ~ObjectName(ParameterName1\1) or ~ObjectName.ParameterName.1 or ~TeLiTabName.ParameterName.1, write the content of an object but only case 1 of ParameterName1. Useful for writing multi-case values. these values might have a single value but still are presented in the table of a TeLiTab (being a multi-value parameter~ParameterName;
   d. #ParameterName#, this method is useful for producing a flow of text and replacing parts in text. #ParameterName# is replaced without leading and trailing spaces. So please realise that "#Pi|F12.3#" (Pi using fixed format, a cell with of 12 and 3 decimal places) returns " 3.142" and is put in the text without spaces, thus as "3.142";
   e. #Boolean Expression[Label]#Text to be placed[Label], use an expression and place the text (or parameter) specified between the labels when the expression is true. Will not work with Objects in a Boolean Expression and functions referring to data in an any data slot;
9. TEMPLATE$ makes it is also possible to generate FORTRAN-look alike formatted data, e.g. for programs requiring an input file containing a formatted set of values. In this case the parameters and their formats are providedwithout "~" or "#" but simply separated by comma's and/or spaces. This mode can be used in individual lines of Template$ when they ONLY contain these formats, so **not** mixed with other text. See the example below for more details
10. Moreover, it is possible to place values in combination with expression (see an example below);
11. Besides the replacement options, there are some options for defining formats. In general the format are defined by ParName|Format$. The formats defined in FORMAT$() can also be used for the TEMPLATE$() function (but with "|" behind the parameter name instead of between quotes).
12. Note that using the format statements a picture connected to a parameter value can be shown (see also Image management, Data use and management and Documentation of knowledge). Use <parametername>|P;
13. Moreover, when @EQEXPLAIN is used, without the format statements, the actual value for a parameter is shown in the template. If you want to use the explanation as shown using @EQEXPLAIN use <parametername>|C;
14. Please note that you can add a Carriage return Linefeed (CrLf) to values by adding "/|" after the format. Example: OWPOL(NT|I18|X2|1)/| will write the first case of NT (indicated by |1) in object OWPOL as integer in a cell of 18 positions followed by two spaces (X2) and than place a Carriage return Linefeed
15. Special addition that will only work in the TEMPLATE$ function:
    a. Adding &Number to the fixed format (F) will round values on multitude of Number. Example: "#Pi|F12.3&0.25" returns " 3.25" and is put in the text without spaces, thus as "3.25"
    b. Integer values can also be printed as words using &English, however, **only for 0-10** and in **English**. Example: "#NPROP|I2&English#" returns "One" for NPROP=1
       In addtion to &English for returning "&Value" you can use either:
       i. &english returning "value"
       ii. &ENGLISH returning "VALUE"
    c. A counter can be included using #&# in the template. Counting starts with 1. Example "#&#) Woord, #&#) Polyestre, #&#) Foam" returns "1) Wood, 2) Polyester, 3) Foam"

       i. Multiple sets of **independent** counters can be used in one template using a specific number of "&" per per counter, i.e. for two independent counters use respectively #&#" for the first one and "#&&#" for the second (independent) counter.

ii. Instead of integer counters, you can use **english words** (first to tenth) by using "%" instead of "&". For independent counters (same mechanism as counters above applies).

Examples

The writing mechanism with "~Name"

Mode%=0

Assume LPP=33.37 with the cell width in de knowledge base defined as 6

TEMPLATE$("Length between PP: ~LPP [m]", 0, LPP)

Steps taken bij Quaestor:

1. Remove ~LPP:

   Length between PP: [m]
2. Write value "33.37 " from position ~ :

   Length between PP: 33.37

Result:
"Length between PP: 33.37"

In this case, insufficient spaces are left in the template between ~LPP and [m].

Mode%=1

To obtain a better result, you can used the ReplaceMode%=1:

TEMPLATE$("Length between PP: ~LPP [m]", 1, LPP)

Steps taken bij Quaestor:

1. Overwrite ~LPP with spaces: Length between PP: [m]
2. Write value "33.37 " from position ~: Length between PP: 33.37 m]

Result:

"Length between PP: 33.37 m]"

Note that 33.37 is followed by a space causing the left bracket to disappear...

Mode%=2

The final option is to use ReplaceMode%=2:

TEMPLATE$("Length between PP: ~LPP [m]", 2, LPP)

Steps taken bij Quaestor:

1. Remove ~LPP: Length between PP: [m]
2. Insert value "33.37 " from position ~: Length between PP: 33.37 [m] Result:

"Length between PP: 33.37 [m]"

Whether to use ReplaceMode%=0,1 or 2 depends on the purpose, the length of the parameter names, the definition of Cell width in the Slot & Properties and the format definition of the values in the template. All options have their advantages and disadvantages....

The writing mechanism with "~ParName" as object

Assume an object GEOM to contain the following data:

0
3 "RADIUS" "THICKNESS" "CHORD"
"1" 0.250 54.0 495.4
"2" 0.450 40.2 520.4
"3" 0.600 21.0 543.7
"4" 0.800 17.0 390.4
"5" 1.000 Â 9.0 Â 14.5

relation:

TEMPLATE$("Thickness distribution:
 Radius Thickness
~GEOM(RADIUS,THICKNESS)", 0, @GEOM)

returns:

"Thickness distribution:
 Radius Thickness
 0.25 54.00
 0.45 40.20
 0.60 21.00
 0.80 17.00
 1.00 9.00"

Values of RADIUS and THICKNESS are formatted according to the properties in the knowledge base, so fixed format with two decimals placed right in a cell of 9 spaces.

relation:

TEMPLATE$(
"Blade root radius : ~GEOM(RADIUS\1) r/R
BlBlade root thisckness : ~GEOM(THICKNESS\1) mm
Tip thickness : ~GEOM(THICKNESS\5) mm", 0, @GEOM)

returns:

"Blade root radius : 0.25 r/R
BlBlade root thisckness : 54.00 mm
Tip thickness : 9.00 mm"

The writing mechanism with "#ParName#"

Assume LPP=33.37 with the cell width in de knowledge base defined as 6

TEMPLATE$("Length between PP: #LPP# [m]", 0, LPP)

Steps taken bij Quaestor:

1. Remove #LPP#: Length between PP: [m]
2. Insert value "33.25" from former starting position of #LPP#: Length between PP: 33.25 [m]

Result:

"Length between PP: 33.25 [m]"

The writing mechanism with "#Parname|Format$#"

Assume LPP=33.37 with the cell width in de knowledge base defined as 6

TEMPLATE$("Length between PP: #LPP|E8.3# [m]", 0, LPP)

Result:

"Length between PP: 0.334E+02 [m]"

The writing mechanism with "#Boolean Expression[Label]#Text to be placed[Label]"

TEMPLATE$("The model will be fitted with#NPROP=2[1]# two propellers and[1]#NPROP=1[1]# one propeller and[1] all appendages", 0, NPROP)

NPROP will be determined by Queastor because it is a parameter in this relation (InpVar).

This relation returns for NPROP=1 :

"The model will be fitted with one propeller and all appendages"

and returns for NPROP=2 :

"The model will be fitted with two propellers and all appendages"

And for NPROP=0 ,

"The model will be fitted with all appendages"

is returned.

In this example for both bolean actions the same label [1] is used. In "Text to be placed" further labels and nested logical constructs are allowed. However, Please note that [Label] in nested constructs should be different from [Label] of the surrounding construct. Using following numbers to label parts is a very convenient way.

The writing mechanism with nested "#Boolean Expression[Label]#Text to be placed [Label]"

TEMPLATE$("The model will be fitted with#NPROP>0[1]##NPROP=2[2]# two propellers and[2]#NPROP=1[2]# one propeller and[2][1] all appendages", 0, NPROP)

This relation will give the same result as the previous example but now with a nested expression for NPROP.

The writing mechanisme for integers as words

Integer values can also be printed as words, however, **only for 0-10**:

TEMPLATE$("#NPROP|I2&English# propeller#NPROP><1[1]#s[1] fitted.", 0, NPROP)

Returns for

NPROP=0(1)2

respectively:

"Zero propellers fitted."
"One propeller fitted."
"Two propellers fitted."

For NPROP=12

"No numeral {only 0-10} for: 12 propellers fitted."

is returned

The writing mechnisme for counters

A counter can be included in Template$ by "#&#", for example

TEMPLATE$("#&#) Wood
#Option1[1]##&#) Polyester
[1]#&#) Foam", 0, Option)

returns for Option=0:

"1) Wood
2) Polyester
3) Foam"

and for Option=1:

"1) Wood
2) Foam"

Instead of integer counters, you can use english words first-tenth by using "%" instead of "&". For independent counters, the same principle is applied as above.

TEMPLATE$("#%#) Wood, #%%#) Polyester and #%%%#) Foam", 0)

returns:

"first) Wood, first) Polyester and first) Foam"

You see that Wood, Polyester and Foam all start with first because they all have independent counters...

The writing mechanism without "~" or "#"

Simply seperated by comma's and/or spaces is described per line of the following part of an existing input template:

POT|A1|X2, RP|A1|X2, RE|A1|X2, PR|A1|X2, ITTC78|A1|X2, EX|A1|X2
DSNR|I18|X2, OVERL|A1, NKOL1|I4, NKOL2|I4, IUNITS|I4/|
OWPOL(NT|I18|X2|1), OWDAT(NP|I18|X2|1)/|
|M4|TEMPOWT|E18.11|X2, FAF|E18.11|X2, PGOE|E18.11|X2, C75D|E18.11|X2
OWPOL(KQPC|E18.11|X2|6), OWPOL(KTPC|E18.11|X2|6), OWDAT(SCJ|E18.11|X2|30)/|

Line 1:

POT|A1|X2, RP|A1|X2, RE|A1|X2, PR|A1|X2, ITTC78|A1|X2, EX|A1|X2

Write POT as a string of one position (A1) followed by 2 spaces (X2),

the same applies for RE, PR, IIT78 and EX

Line 2:DSNR|I18|X2, OVERL|A1, NKOL1|I4, NKOL2|I4, IUNITS|I4/|

Write DSNR as an integer value in a cell of 18 positions (I18) followed by two spaces (X2),

Write OVERL as a string of one position

Write NKOL1 as an integer value in a cell of 4 positions (I4)

Write IUNITS as integer value in a cell of 4 positions followed by a Carriage return Linefeed (CrLf) as indicated by "/|" after the format.

Line 3:

OWPOL(NT|I18|X2|1), OWDAT(NP|I18|X2|1)/|

Write NT element 1 in object OWPOL as integer in a cell of 18 positions followed by two spaces (X2).

If object OWPOL is not present in the solution there are two options:

a. The data slot of OWPOL contains a @DBDEFAULT, e.g.: @DBDEFAULT :0 (IT IS FOR OBJECTS ONLY ALLOWED TO USE ")" as DBDEFAULT)
   In that case, all OWPOL formats are presenting a value of 0.
b. The data slot of OWPOL does not contain a @DBDEFAULT In that case, all OWPOL formats are presenting a value of 0.

If in a format OWPOL(NT|I18|X2|1) OWPOL exists and NT does neither exist in OWPOL nor in the level(s) above, the @DBDEFAULT value of NT is printed if available or the message:

"NT has no DBDEFAULT"

Idem NP , however followed by CrLf as indicated by "/|" after the format

Line 4:

|M4|TEMPOWT|E18.11|X2, FAF|E18.11|X2, PGOE|E18.11|X2, C75D|E18.11|X2|M4| is the line counter: each time when 4 values have been written, write a CrLf.

Write TEMPOWT in E format using 11 decimals in a cell of 18 length (E18.11), followed by two spaces (X2).

Idem for FAF , PGOE and C75D .C75D is fourth value from |M4| so write CrLf.

Line 5:

OWPOL(KQPC|E18.11|X2|6), OWPOL(KTPC|E18.11|X2|6), OWDAT(SCJ|E18.11|X2|30)/|

write 6 subsequent values of OWPOL(KQPC) with an E18.11 format followed by two spaces (X2), each time when the line counter |M4| reaches 4, a CrLf is written. If OWPOL contains 4 instead of the indicated 6 values, the 5th and 6th values are given the value zero.

Idem OWPOL(KTPC|E18.11|X2|6)

Idem OWDAT(SCJ|E18.11|X2|30) , expect 30 cases in table OWDAT, if it contains fewer values, write missing values as 0.

After the 30 values, write a CrLf as indicated by "/|"

Writing parameters being part of a TeLiTab

A parameter being part of a TeliTab cannot be written as ~ParName|Format. In that case it should be either:

1. ParName|Format
2. #ParName|Format#.

Assuming a TeLiTab value of Obj$ of:

```
0
3 "RADIUS" "THICKNESS" "CHORD"
"1" 0.250 54.0 495.4
"2" 0.450 40.2 520.4
"3" 0.600 21.0 543.7
"4" 0.800 17.0 390.4
"5" 1.000 9.1 14.5
```

```
TEMPLATE$(
"2nd thickness THICKNESS.2|F6.2/|
3rd thickness: #THICKNESS.3|F6.2# [mm]
4th thickness: ~THICKNESS.4|F6.1
5th thickness: ~THICKNESS.5", 0, Obj$)
```

returns:

```
" 40.20
3rd thickness: 21.00 [mm]
String: "4th thickness: ~THICKNESS.4" cannot be Fixed (F) formatted
5th thickness: 9.1"
```

Of the first line: "2nd thickness " is not included in the result because additional, non formatted string information is omitted when you use this definition, so only

THICKNESS.2|F6.2/|

is returned, being:

```
" 40.20
"
```

(including a CrLf)

The second line returns

"3rd thickness: 21.00 [mm]"

in which 21.00 is a trimmed F6.2 of 21

The third line includes an error message:

"String: "4th thickness: ~THICKNESS.4" cannot be Fixed (F) formatted"

because the total string

"4th thickness: ~THICKNESS.4"

is viewed as the value to be formatted F6.1

The last line returns

"5th thickness: 9.1"

The value 9.1 is the FF2 formatted value from which decimal closing zero's are trimmed.

Writing parameters being part of an object

An object can be treated like a TeLiTab.

if DATA is an object containing:

```
1
"DATA"
{
  0
    2   "A" "B"
   "1"    1   2
   "2"    2   4
   "3"    3   6
   "4"    4   8
   "5"    5  10
   "6"    6  12
   "7"    7  14
   "8"    8  16
   "9"    9  18
   "10"  10  20
}
```

Then relation:

```
TEMPLATE$("An example:
~DATA(A\3)
~DATA(B)",0,@DATA)
```

returns:

```
An example:
3.00
     2.00
     4.00
     6.00
     8.00
    10.00
    12.00
    14.00
    16.00
    18.00
    20.00
```

The third case of A is taken followed by the content for B in the object. For both parameter no format decription is used (so the format of the Workbase/Slots & Properties is taken).

You can use format tags in the script:

```
TEMPLATE$("An example:
~DATA(A\3)
~DATA(B|F12.3/|)",0,@DATA)
```

returns:

```
An example:
3.00
       2.000
       4.000
       6.000
       8.000
      10.000
      12.000
      14.000
      16.000
      18.000
      20.000
```

The third case of A is taken without any format decription (so the format of the Workbase/Slots & Properties is taken). this is followed by the content for B in the object using a Cell width of 12 and 3 decimal places.
Please note, when you use format statement for paramters in the object, you have to close with a CrLf, thus "/|". otherwise all values will be placed behind each other:

```
TEMPLATE$("An example:
~DATA(A\3)
~DATA(B|F12.3)",0,@DATA)
```

returns:

```
An example:
3.00
     2.000     4.000     6.000     8.000    10.000    12.000    14.000    16.000    18.000    20.000
```

Writing values using expressions

By placing expressions between brackets, you can first perform calculation of which the result will be placed:

```
"Averaged draught trial condition      [m]   = ~(TFOR_INT/2+TAFT_INT/2) "
```

TFOR_INT and TAFT_INT will first be used to calculate the average draught.

or

```
TEMPLATE$("Company : #COMPANY$#
Street & City: #STREET$ + CHR$(13) + CHR$(10) + SPACE$(15) + CITY$|#", 0, DEPT$, COMPANY$,
STREET$, CITY$)
```

returns for:

```
COMPANY$= "Qnowledge"

STREET$="Haagsteeg 2"

CITY$="Wageningen"

Company : Qnowledge
Street & City: Haagsteeg 2
Wageningen
```

The expression "STREET$ + CHR$(13) + CHR$(10) + SPACE$(15) + CITY$" is surrounded by "#" and terminated by "|".

Quick links: Functions overview | Attribute overview | Constants overview | Dimensions overview