PARSE#

PARSE# returns a Telitab set parsed from a document

Syntax

PARSE#(Document$,[InpVar])

Arguments

- Document$ is a string value containing either a Telitab set or a (set of) value(s);
- InpVar are optional parameters or objects that should be computerd prior to the parsing.

Remarks

1. The parsing recipe should be included in the **data slot** of the expression in which the PARSE#() function is used.
2. If the data slot the relation does **not** contain any filter data, the program seeks the filter in the data slot of parameter Document$.
3. The parse scripts are best explained by the examples below. Their are however several issues to take into account:
    a. A parse script starts with @FILTER and ends with @ENDFILTER;
    b. A list value is identified in the script by @LIST(InpVar), connecting the parsed value or string to InpVar (the parameter you want to connect the parsed information to), followed by the exact text part to identify the information including the indicators on the possitions of the value or string to be parsed (see 3. to 5. below);
    c. For parsing a value use "vv" (without the quotes);
    d. For parsing a string value use "ss";
    e. (version 2.47.2 and later) for parsing any value or **string values including spaces**, use "xx";
    f. **Please note that a minimum of two characters for "v" and "s" are required in the filter;**
    g. A table is identified in the script by @TABLE(InpVar), connecting the parsed values and/or strings to InpVar (the parameters you want to connect the parsed table to), followed by the exact sequence of indicators on the possitions of the values or strings to be parsed as table (see 3. to 5. above). **Note that, although you might only want to use one collumn out of a table, you need to give the indicators for all values in the table. Otherwise Quaestor will not be able to recognise the connection between the specified script and the data**;
    h. Moreover, realise when using "ss" or "vv" Quaestor expects a full table with values. When values in the table are missing, the whole row will be ignored. To avoid this, you can use "aa" instead. **When using "aa" all missing date is assumed to be pending.**
    i. Please make sure that @LIST() and @TABLE() are directly followed by a CrLf (an "enter").
    j. Use @TELITAB(InpVar) and @ENDTELITAB(InpVar) to create a separate TeLiTab (or Object) connected to InpVar (the parameter /object you want to connect the parsed set of information to), in combination with @LIST() and @TABLE() to parse the information for this TeLiTab;
    k. @FILTER can included conditions (see also the example below). Syntax: @FILTER(Condition), with Condition one or more expressions including parameters seperated by comma's; Please be aware that both the parameters an their determined values should be available (so include them to your function syntax).
4. In some cases, it might be necessary to prepare and modify the document in order to comply with the rules above. In almost all cases this is possible by means of smart use of text manipulation functions like REPLINS$, PARAGRAPH$ and SECTION$ (see also String document functions). By means of PARAGRAPH$ and SECTION$ you can select specific areas out of a document to make sure that an unique filter can be created for that part. REPLINS$ can be used to remove things like CrLf's ("enters"), spaces, part of names, etc.).

Examples

First example

Let Document$ contain the following data.

"
MODEL PROPULSION FACTORS

PROPULSION TEST NO. 45482 DRAUGHT FWD 3.95 M
OPEN WATER TEST NO. 37816 DRAUGHT AFT 4.13 M
SHIP MODEL NO. 1234
PROPELLER MODELS NO. 5165 R+L

VS VM NM FN KT KQ KQ-O J WT ETA-O ETA-R

14.0 1.878 10.347 .237 .1390 .02643 .02590 .800 .101 .683 .980
15.0 2.013 11.141 .254 .1401 .02657 .02605 .798 .099 .683 .980
16.0 2.147 12.032 .271 .1445 .02717 .02666 .789 .097 .681 .981
17.0 2.281 12.978 .288 .1500 .02792 .02742 .779 .096 .678 .982
18.0 2.415 13.959 .305 .1557 .02870 .02820 .768 .094 .675 .983
19.0 2.549 14.977 .322 .1616 .02951 .02901 .756 .093 .671 .983

NOTES:-FOR EXPLANATION OF ABBREVIATIONS SEE LIST OF SYMBOLS
-THE FACTORS ABOVE ARE MODEL VALUES AT THE SELF PROPULSION POINT
OF SHIP CORRESPONDING TO A FRICTION CORRECTION DETERMINED BY
THE ITTC-1957 FORMULA,
WITH A FORM FACTOR OF 1+K= 1.200,
A MODEL-SHIP CORRELATION ALLOWANCE OF CA= .00070
AND A TEMPERATURE OF THE TANK WATER OF 16.4 DEGREES C.
-THE PROPULSION FACTORS ARE BASED ON THRUST IDENTITY
"

And there is a relation

ParseDocument# = PARSE#(Document$)

With the following information in the data slot.

@FILTER
@TELITAB(OpenWater)
@LIST(TestNr)
OPEN WATER TEST NO. vvvvv
@TABLE(VS VM NM FN KT KQ KQ-0 J WT ETA-O ETA-R)
vvvv vvvvv vvvvvv vvvv vvvvv vvvvvv vvvvvv vvvv vvvv vvvv vvvv
@ENDTELITAB(OpenWater)
@ENDFILTER

Explanation

@TELITAB(OpenWater)

opens object "OpenWater":

1
"OpenWater"
{

Then

@LIST(TestNr)
OPEN WATER TEST NO. vvvvv

makes a single Telitab value of parameter "TestNr" on the basis of the filter definition:

"TestNr" 37816

And then

@TABLE(VS VM NM FN KT KQ KQ-O J WT ETA-O ETA-R)
vvvv vvvvv vvvvvv vvvvv vvvvvv vvvvvvv vvvvvvv vvvvv vvvvv vvvvv vvvvv

creates a table header:

10 "VS" "VM" "NM" "FN" "KT" "KQ" "KQ-O" "J" "WT" "ETA-O" "ETA-R"

and the filter definition:

vvvv vvvvv vvvvvv vvvvv vvvvvv vvvvvvv vvvvvvv vvvvv vvvvv vvvvv vvvvv

seeks lines in the source document that are containing values in **ALL** cells corresponding with the locations marked by v's and are added to the parsed result.
And finally the object is closest using:

@ENDTELITAB(OpenWater)

Creating

}

The combined result is:

```
1
   "OpenWater"
   {
   1
   "TestNr" 37816
   10 "VS" "VM" "NM" "FN" "KT" "KQ" "KQ-O" "J" "WT" "ETA-O" "ETA-R"
   "1" 14.0 1.878 10.347 .237 .1390 .02643 .02590 .800 .101 .683 .980
   "2" 15.0 2.013 11.141 .254 .1401 .02657 .02605 .798 .099 .683 .980
   "3" 16.0 2.147 12.032 .271 .1445 .02717 .02666 .789 .097 .681 .981
   "4" 17.0 2.281 12.978 .288 .1500 .02792 .02742 .779 .096 .678 .982
   "5" 18.0 2.415 13.959 .305 .1557 .02870 .02820 .768 .094 .675 .983
   "6" 19.0 2.549 14.977 .322 .1616 .02951 .02901 .756 .093 .671 .983
   }
```

Second example

Assume the following document to be parsed:

```
Speed      Power      Power    Rate of   Rate of   Prop.
                        Revs.    Revs.   Diam.
     kn        kw        hp       rps       rpm       m

    10.00      19.30      26.24      8.66     519.5     0.60    CARE
    12.00      65.56      89.16     12.16     729.9     0.60    CARE
    14.00      80.03     108.84     13.33     799.7     0.60    CARE
    16.00      92.31     125.54     14.36     861.5     0.60
    18.00     104.07     141.53     15.32     919.5     0.60
    20.00     118.47     161.12     16.39     983.2     0.60
    22.00     136.15     185.16     17.51    1050.3     0.60
    24.00     159.04     216.30     18.74    1124.3     0.60
    26.00     187.54     255.05     20.08    1204.6     0.60
    28.00     222.11     302.07     21.52    1291.4     0.60
    30.00     265.46     361.02     23.09    1385.4     0.60    CARE
    32.00     322.98     439.25     24.87    1492.4     0.60    CARE
    34.00     395.97     538.51     26.85    1610.9     0.60    CARE
```

You see that note CARE behind the data. The only way to parse this data is to use the "aa" mask:

```
@FILTER
@TABLE(VS_kts PD PD_hp N_s N DIAM Warning$)
 vvvvvvvvv vvvvvvvvvv vvvvvvvvvv vvvvvvvv vvvvvvvv vvvvvvv aaaaaaa
@ENDFILTER
```

Would you have used the "ss" mask instead of the "aa" mask the result would be:

```
0
7   "VS_kts"   "PD"   "PD_hp"   "N_s"   "N"   "DIAM"   "Warning$"
"1"   10.00      19.30      26.24      8.66     519.5     0.60    CARE
"2"   12.00      65.56      89.16     12.16     729.9     0.60    CARE
"3"   14.00      80.03     108.84     13.33     799.7     0.60    CARE
"11"  30.00     265.46     361.02     23.09    1385.4     0.60    CARE
"12"  32.00     322.98     439.25     24.87    1492.4     0.60    CARE
"13"  34.00     395.97     538.51     26.85    1610.9     0.60    CARE
```

With "aa" it will be:

```
0
7   "VS_kts"   "PD"   "PD_hp"   "N_s"   "N"   "DIAM"   "Warning$"
"1"   10.00      19.30      26.24      8.66     519.5     0.60    CARE
"2"   12.00      65.56      89.16     12.16     729.9     0.60    CARE
"3"   14.00      80.03     108.84     13.33     799.7     0.60    CARE
"4"   16.00      92.31     125.54     14.36     861.5     0.60 -999999
"5"   18.00     104.07     141.53     15.32     919.5     0.60 -999999
"6"   20.00     118.47     161.12     16.39     983.2     0.60 -999999
"7"   22.00     136.15     185.16     17.51    1050.3     0.60 -999999
"8"   24.00     159.04     216.30     18.74    1124.3     0.60 -999999
"9"   26.00     187.54     255.05     20.08    1204.6     0.60 -999999
"10"  28.00     222.11     302.07     21.52    1291.4     0.60 -999999
"11"  30.00     265.46     361.02     23.09    1385.4     0.60    CARE
"12"  32.00     322.98     439.25     24.87    1492.4     0.60    CARE
"13"  34.00     395.97     538.51     26.85    1610.9     0.60    CARE
```

In which -999999 stands for PENDING values.

Troubleshoot/Tips & Tricks

1. Things can very easily go wrong when you are parsing:
    a. The format of the text in the filter has to be exactly the same as that of the text in the original document. Therefore, always use Courier as font in your text editor in order to have non-scalable characters and thus being able to align spaces.
    b. The vv's and ss's have to be exactly in the right positions, otherwise the function doesn't work.
2. It is a good idea to copy the original text into the data slot of your PARSE# relation when making a filter. In this way you have guidance while placing everything in the right position.
   Here is an example. The table is pasted into the dataslot and the vv's are written underneath to place them in the right position:

   ```
   14.0 1.878 10.347 .237 .1390 .02643 .02590 .800 .101 .683 .980
   vvvv vvvvv vvvvvv vvvv vvvvv vvvvvv vvvvvv vvvv vvvv vvvv vvvv
   ```
3. Non-Telitab parsing.
   For the above source document using the filter:

   ```
   @FILTER
   OPEN WATER TEST NO. vvvvv
   @ENDFILTER
   ```

   the result will be:

   "37816"

   and with

   ```
   @FILTER
   vvvv vvvvv vvvvvv vvvvv vvvvvv vvvvvvv vvvvvvv vvvvv vvvvv vvvvv vvvvv
   @ENDFILTER
   ```

   the result:

   ```
   "14.0 1.878 10.347 .237 .1390 .02643 .02590 .800 .101 .683 .980
   15.0 2.013 11.141 .254 .1401 .02657 .02605 .798 .099 .683 .980
   16.0 2.147 12.032 .271 .1445 .02717 .02666 .789 .097 .681 .981
   17.0 2.281 12.978 .288 .1500 .02792 .02742 .779 .096 .678 .982
   18.0 2.415 13.959 .305 .1557 .02870 .02820 .768 .094 .675 .983
   19.0 2.549 14.977 .322 .1616 .02951 .02901 .756 .093 .671 .983"
   ```

   is obtained
4. Use of conditional Filter
   A Relation or parameter can contain a set of conditional filters:

   ```
   @FILTER(METH2D3D=1,RE="T")
   @TABLE(VS N n PS PE ETA-D TH R THDF)
   vvvvvv vvvvv vvvvvvv vvvvvv vvvvv vvvvvv vvvvvv vvvvv vvvvv
   @ENDFILTER
   @FILTER(METH2D3D=1,RE="F")
   @TABLE(VS N n PS TH)
   vvvvvv vvvvv vvvvvvv vvvvvv vvvvvv
   @ENDFILTER
   @FILTER(METH2D3D=0)
   ```

@TABLE(VS N n PDTANK PS)
vvvvvvv vvvvvv vvvvvv vvvvvv vvvvvv
@ENDFILTER

this means:

@FILTER(METH2D3D=1,RE="T")

means: "If METH2D3D=1 AND RE="T" then apply this filter as closed by @ENDFILTER" for the first @FILTER(Condition) of which condition = True is applied.

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