Control Flow

There are some control flows that allows to define how certain commands are executed.

#LOOP

This control flow allows a simple for-loop, which may be used to execute the same queries or commands two or more times.

Usage

The loop begins with a #LOOP command that needs some parameters and ends with #ENDLOOP. The LOOP command's syntax is as follows

<counter-name> <counter-start> UPTO <counter-end>

where <counter-name> is a string containing a variable. <counter-start> and <counter-end> are integers and defines that the loop starts with <counter-start> (inclusively) and runs until <counter-end> (exclusivly). Therefore, the loop repeats the commands between #LOOP and #ENDLOOP <counter-end> - <counter-start> times. Furthermore, the counter can be used like a variable that was set using #DEFINE.

Example

The examples shows a loop that repeats 10 times (i=0 until i<10) and executes the "#RUNQUERY SELECT ..." accordingly ten times. Furthermore, the counter "i" is used within the query, so that each ${i} is replaced by the current value of i from the loop.

```
#LOOP i 0 UPTO 10
    #RUNQUERY
    SELECT ${i} AS b, * FROM bid
#ENDLOOP
```

You may also use ${i-1} or ${i+1} (only this two!). For example, this

```
#LOOP i 2 UPTO 4
    #RUNQUERY
    SELECT ${i-1} AS a, ${i+1} AS b, * FROM bid WHERE b>${i}
#ENDLOOP
```

is equal to

```
#RUNQUERY
SELECT 1 AS b, * FROM bid3 WHERE b>2
#RUNQUERY
SELECT 2 AS b, * FROM bid4 WHERE b>3
```

You may also use an additional offset variable. This offset variable adds a defined value to the current value of the actual loop variable. Following example uses an offset of x = 5:

```
#RUNQUERY
#LOOP i 2 UPTO 5 WITH x 5
    SELECT ${x} AS b, * FROM bid
#ENDLOOP
```

This is equal to:

```
#RUNQUERY
SELECT 7 AS b, * FROM bid
SELECT 8 AS b, * FROM bid
SELECT 9 AS b, * FROM bid
```

#FOREACH_IN

You can also define a foreach loop, e.g., as follows:

```
#LOOP x FOREACH_IN 1,2,...,9,10,20,...,60,90,120
/// Some content
#ENDLOOP
```

#IFDEF/#IFNDEF

With #IFDEF it is possible to check whether a variable exists and was set by #DEFINE or not. This is useful, for example, to run certain queries corresponding to the current setting. Use #IFNDEF for case where to check if a variable has no value

Usage

Like common if-statements in other programming languages, the structure of #IFDEF follows "if-then-else", where the "else" part is not necessary. Accordingly, the structure is as follows:

#IFDEF <variable-name>

<then-commands>

#ENDIF

or

#IFDEF <variable-name>

<then-commands>

#ELSE

<else-commands>

#ENDIF

So, if the variable named <variable-name> exists, the control flow runs the <then-commands>, if not and there is an else-part, the <else-commands> are executed. Look also add #DEFINE and #UNDEF to set or unset variables.

Example

The example defines a variable called latencyOn and uses the #IFDEF command to use either StandardLatency for the transformation configuration, if latencyOn is set or Standard if it is not set. Obviously, this example uses allways the <then-command>-part (since latencyOn is set), so you may switch to another transformation config by simply commenting the #DEFINE command out so that the <else-commands> are used.

```
#DEFINE latencyOn
....
#IFDEF latencyOn
        #TRANSCFG StandardLatency
#ELSE
        #TRANSCFG Standard
#ENDIF
```

#IFSRCDEF/#IFSRCNDEF

Similar to #IFDEF but checking if a source with the given name is registered in the data dictionary. This can be helpful to define sources only if they are not already defined

```
#IFSRCNDEF basestream
#INCLUDE ${WORKSPACEPROJECT/}Source.qry
#ENDIF
```

#IFQDEF/#IFQNDEF

Similar to #IFDEF but checking if a query with the given name is registered in the executor (see command QNAME) . This can be helpful to define queries only if they are not already defined

```
#IFQNDEF myQuery
#QNAME myQuery
#RUNQUERY
<query text>
#ENDIF
```

#IF

Similar to #IFDEF and #IFSRCDEF but checking an arbitrary expression. This can be helpful to execute a block only if the expression evaluates to true.

```
#IF toInteger(CPU) > 1
#RUNQUERY
...
#ENDIF

...

#DEFINE a_string FOO
#IF a_string == "FOO"
...
#ENDIF
```

#LOOP radius FOREACH_IN 5000,10000