Extended syntax for the expressions

The extended syntax for the expressions (statements) of eval tags and calculated historical values

The following chapter describes the syntax that is usable in the expressions of objects of *Eval tag* type and the statements of objects of *Historical value* type.

Initialization part

The initialization part must always start with the **INIT** keyword. This part may contain:

- declarations of local variables,
- labels,
- **GOTO** *label:* commands,
- **IF** *condition* **GOTO** *label:* commands
    - the label must be always ended by the character ":" (as well as a label in **GOTO** commands),
- commands based on conditional evaluation,
  IF *condition1* **THEN**
  action1
  ELSIF *condition2* **THEN**
  action2
  ELSIF *condition3* **THEN**
  action3
  .....
  ELSE
  action
  ENDIF
- command **RETURN** ends the calculation and assigns the value to result (constant or expression) after **RETURN** command,
- command **EXIT** causes a move on part **FINALLY**.

The initialization part is optional. If it is not declared, the original syntax is valid.
If the initialization part has been declared, the expressions, determining the value of the object, must be specified after the **FINALLY** keyword.

Limitations

The number of executed rows is limited in D2000 systems. Nowadays, the number cannot be defined or changed at all.
The number (*ExecuteRows*) depends on the type of used process.

For process D2000 Calc:

- *ExecuteRows* > 1000 - the following log: "*EvalTag_Name - executed rows : row_count*" is to be written into the log file of process **D2000 Calc** (*calc.log*)
- *ExecuteRows* > 100 000 - the system variable SystemError acquires the following value: "*EvalTag_Name - infinite loop*"

For process D2000 Archiv:

- *ExecuteRows* > 1000 - the following log: "*HistoricalValue_Name - executed rows : row_count*" is to be written into the log file of process **D2000 Archiv** (*archiv.log*)
- *ExecuteRows* > 10 000 - the system variable SystemError acquires the following value: "*HistoricalValue_Name - infinite loop*"

If the value of a different type is assigned to the system variable, a runtime error occurs while the expression is executing. The runtime error stops the script evaluation (the other rows will not be evaluated) and always returns an *Invalid* value. To display this error, activate "debug runtime errors".

Exceptions:

- mutual conversion *HBJ - Integer*
- mutual conversion *Relative time - Real - Integer*
- direct conversion from *Boolean* to *Relative time*, *Real* or *Integer*

Example 1

Calculation of the sum of the values in column *A2* of the structured variable *SV*. The sum only includes the values of the lines, where the value in column *A1* is greater or equal to 3.

```
INIT

INT _rows
INT _rowNr
REAL _sum

_rows := SV.A\DIM
_sum := 0.0

START:
_rowNr :=1

CYCLE:
IF SV.A[_rowNr]^A1 < 3 GOTO SKIP:
_sum := _sum + SV.A[_rowNr]^A2

SKIP:
_rowNr := _rowNr + 1

IF _rowNr <= _rows GOTO CYCLE:


FINALLY

_sum
```

Example 2

This example shows a sequence number of one-third of the running minute - three ways of value return.

```
INIT

INT _third

IF Sec <= 20 THEN
    RETURN 1
ELSIF Sec <= 40 THEN
   _third := 2
    RETURN _third
ELSE
   _third := 3
    EXIT
ENDIF

FINALLY
_third
```

In both examples, the timestamp of the expression will acquire the time of evaluation of the expression. If the expression is expected to acquire other value than the time of evaluation of the expression, it should be defined in the expression (after FINALLY!).

Example:

```
FINALLY
%NtV(_sum, _time)
```

To define the user flags for the resultant value of the expression, specify the definition of user flag in the expression.

Example:

```
FINALLY
%SetFlags(_sum, @A, _boolean)
```

Related pages:

Historical values
Eval tags