Using Macro Expressions on Parsing or Composing a Flat File

For parsing and composing flat files there is a set of counters and related macros available that provide extended features.

Type	Parse	Compose	Description	ID
automatic counters	✅	✅	Eight automatic counters that can be accessed via their IDs `AUTO0` .. `AUTO7`.	`AUTO0 .. AUTO7`
automatic line counters	✅	❌	Two automatic line counters that can be accessed via their IDs LINE`0` and LINE`1`.	`LINE0, LINE1`
custom counters	✅	✅	Unlimited custom counters that can be accessed via their IDs `CUSTOM0` .. `CUSTOMx`.	`CUSTOM0 .. CUSTOMx`

You cannot access these counters directly but only via the macro expressions listed in the table below. Macro expressions referring to one of the counters can be specified on flat file **records** and on flat file **properties** (see also Defining a Flat File Data Structure) in the stereotype attributes **parseMacro** and **composeMAcro**.

- When parsing/composing the first record, all counters have the initial value of 0.
- Automatic counters automatically increase with each processed record.
- Custom counters have to be increased manually using the **IncreaseCounter**() macro (see below).

The counters can be handled using the following macros:

Macro	Available on	Available for	Description	Example

ResetCounter(`ID`[, *Value*])	Record Property	`AUTO` `CUSTOM`	Reset the counter `ID` to 0 or a given *Value*.	`ResetCounter (AUTO0)`
				
				 `ResetCounter (AUTO0, 1)`

IncreaseCounter(`ID`)	Record Property	`AUTO` `CUSTOM`	Increases the counter `ID` by 1.	`IncreaseCounter (CUSTOM2)`
GetCounter(`ID`)	Property	`AUTO` `LINE` `CUSTOM`	Read the value of a counter and store it to the current attribute.	`GetCounter(LINE0)`
VerifyCounter(`ID`)	Property	`AUTO` `LINE` `CUSTOM`	Compare the value of a counter with the current attribute. This macro will throw an exception if the values are not equal.	`VerifyCounter (CUSTOM0)`

> ⓘ For **GetCounter()** and **VerifyCounter()** only attributes of type **Integer** are supported.

Example

The Flat File adapter example shows the usage of an automatic counter to provide a line number to the data records during the composing process. Refer to Defining a Flat File Data Structure for more details on the example flat file.

- Property **line** stores the number of the product line by using `GetCounter(AUTO0)`.
- With every **Header** record, the counter gets reset to 0 by using `ResetCounter(AUTO0)`. This way the header record will not be counted.

The picture below shows the composing process for the example with emphasis on macro execution.

open file → fetch

<<fileflow>>
ProductList

Note: Reset counter to 0

check

record
evaluationOrder = 1
execute macro

<<FixedFileRecord>>
Header
(attribute.layout = fixed,
composeMacro = ResetCounter(AUTO0, 0),
evaluationOrder = 1,
lineNumber = 1)

0 1 **property**

+headerLine: String (external.length="75")

next

check

Note: Set line field from counter

record
evaluationOrder = 2

<<FixedFileRecord>>
Product
(attribute.layout = fixed,
evaluationOrder = 2)

property
execute macro

0..*
+line : String (composeMacro=GetCounter(AUTO0), external.length="5", order="10")
+number : String (external.length="5", order="20")
+name : String (external.length="20", order="30")
<<FixedFileAttribute>> +category : String (external.length="10", order="40")
+type : String (external.length="10", order="50", padding="Left[" "]")
<<FixedFileAttribute>> +price : Float (external.length="10", order="60", format="FM0000000.00, padding="Left["0"]")
<<FixedFileAttribute>> +semicolInterval : Integer (external.length="5", order="60", padding="Left[" "]")

next

check → out

out

out