**3GPP TSG-SA5 Meeting #145-e *S5-225476***

**e-meeting, 15th Aug 2022 - 24th Aug 2022**

**Source: Nokia, Nokia Shanghai Bell**

**Title: pCR 28.831 Add simple XPath profiles**

**Document for: Approval**

**Agenda Item: 6.8.2.4 - FS\_eSBMAe\_WoP#4**

# 1 Decision/action requested

***The group is requested to discuss and approve the pCR below***

# 2 References

[1] 3GPP TS 28.831: " Management and orchestration; Study on basic Service-Based Management Architecture (SBMA) enabler enhancements"

# 3 Rationale

None.

# 4 Detailed proposal

The following changes are proposed for TR 28.831[1].

|  |
| --- |
| **Begin of modifications** |

##### 4.2.5.2.4 XPath 1.0 profiles

*Editor's note: The content of this clause is work in progress and subject to change.*

An XPath expression matches the production "Expr" defined in XPath 1.0 [2], clause 3.1. However, the general expression is much too generic for selecting nodes of an input document. For rexample, it allows also expression like

(5, 256)[2]

which selects the second item in the sequence (5, 256), hence 256. This expression does not work at all on an input document. Even if "5" and "256" is replaced by some XPath expression evaluating to numbers, the expression cannot be used for selecting nodes.

This is why only one or more XPath profiles are needed.

Two XPath profiles are proposed:

* Basic profile: Supporting limited features allowing XPath to browse the document from one element node to another. The XPath EBNF is detailed in annex A.1. In this profile a location path is defined as an absolute location path. An absolute location path consists of a sequence of one or more location steps separated by / and preceded by /. The location steps in an absolute location path are composed together from left to right. Each step-in turn selects a set of nodes relative to a context node. Note that a / by itself selects the root node of the document. The basic profile supports a predicate that filters on the "id".

Example: /SubNetwork[id="SN1"]/ManagedElement[id="ME2"]/attributes

A location step is composed of

* + an axis specifier, which specifies the tree relationship between the nodes selected by the step and the context node
	+ a node name which specifies the node name of the node selected by the location step. The node name can be a wildcard "\*".

The axis specifier includes two axes:

* + Child: axis containing the children of the context node

Example (unabbreviated syntax): /child::SubNetwork/child::\*

Example (abbreviated syntax): /SubNetwork/\*

* + Descendant: axis containing the descendants of the context node; a descendant is a child or a child of a child and so on

Example: /SubNetwork/descendant::\*

The predicate

* + is an equality expression with the "=" operator, the relative location path "id" on the left side and a literal string on the right side.

Note that the axis specifier "child::" can be omitted from a location step, because child is the default axis. For example, a location path /SubNetwork/ManagedElement is short for /child::SubNetwork/child::ManagedElement.

Regarding the asterisk character "\*", it is reserved to denote a wildcard when used in the location path. It selects all element children of the context node. In the the EBNF notation, the character "\*" mentions repetition symbol (it can be also represented inside curly brackets followed by the "\*").

* Advanced profile: supporting more advanced features like the usage of the predicates. The XPath EBNF is detailed in annex A.2. In this profile, a relative location path is added as a second option to the location path. A relative location path consists of a sequence of one or more location steps separated by / and it does not need to start from the root node as the absolute path.

Example: / / attributes

In this profile, in addition to the axis specifier and Node Name (same as the basic profile), a more sophisticated predicate option is added to the location step. The predicate uses arbitrary expressions to further refine the set of nodes selected by the step. Predicates are defined by an expression (PredicateExpr) always embedded in square brackets. A PredicateExpr can be as follows:

* + An EqualityExpr: expression evaluated by comparing the objects that result from evaluating the two operands. On the right side, the operand is a literal string object and on the left side a PathExpr is introduced as an operand. The PathExpr might be a location path returning an object having one of the following types: node-set, boolean, number or string.

If the PathExpr object is a node-set, then each node in the node-set is compared to the literal string defined in the right side. Note that a node in the node-set might have one/many descendants nodes. And in such a case the comparison expression is evaluated to "false". This kind of comparison is accepted by Xpath (correct syntax); and it returns an empty result. Unfortunately, in the EBNF the PathExpr can not be constrained to avoid such a comparison (See more details on handling comparison based on the node-set in clause 3.4 [2]).

Examples: /SubNetwork/ManagedElement[id="ME1"]

* + InEqualityExpr: expression defining a comparison of a PathExpr object to a given number. Here again, the same problem related to node-set comparison as discussed above is true. Note that here object to be compared is converted to a number as if by applying the number function (more details in clause 3.5 [2]).

Example: //attributes/ThresholdLevels[level>1]

* + A functionCall: evaluated by using the Function Name to identify a function in XPath function librairy. Each function in the function library is specified using a function prototype, which gives the return type, the name of the function, and the type of the arguments (more details in Clause 4[2]).

Examples:

/SubNetwork/ManagedElement[starts-with(id,"ME2")]

/SubNetwork/ManagedElement[contains(id,"ME")]

The rest of the grammar introduced in the EBNF defines additional rules for the lexical structure (Literal, Number and Name structure). Note also that some functions are excluded since they are not useful for this profile.

|  |
| --- |
| **Next modification** |

# Annex A

*Editor's note: The content of this clause is work in progress and subject to change.*

##### A.1 EBNF for basic XPath profile

LocationPath ::= AbsoluteLocationPath

AbsoluteLocationPath ::= Step

 | AbsoluteLocationPath '/' Step

Step ::= AxisSpecifier NodeName Predicate\*

AxisSpecifier ::= AxisName '::'

 |

AxisName ::= 'descendant'

| 'child'NodeName ::= '\*'

| Name

Predicate ::= '[' 'id' '=' Literal ']'

Literal ::= '"' [^"]\* '"'

| "'" [^']\* "'"Name ::= NameStartChar (NameChar)\*

NameChar ::= NameStartChar | "-" | "." | [0-9]

NameStartChar ::= [A-Z] | "\_" | [a-z]

##### A.2 EBNF for advanced XPath profile

LocationPath ::= RelativeLocationPath

| AbsoluteLocationPath

AbsoluteLocationPath ::= '/' RelativeLocationPath?

| '//' RelativeLocationPath

RelativeLocationPath ::= Step

| RelativeLocationPath '/' Step

| RelativeLocationPath '//' Step

Step ::= AxisSpecifier NodeName Predicate\*

AxisSpecifier ::= AxisName '::'

 |

AxisName ::= | 'descendant'

| 'child'

NodeName ::= '\*'

| Name

Predicate ::= '[' PredicateExpr ']'

PredicateExpr ::= EqualityExpr

 | InEqualityExpr

 | PathExpr

EqualityExpr ::= PathExpr '=' Literal

InEqualityExpr ::= PathExpr '<' Number

| PathExpr '>' Number

| PathExpr '<=' Number

| PathExpr '>=' Number

PathExpr ::= LocationPath

| FunctionCall

FunctionCall ::= FunctionName '(' ( Argument ( ',' Argument )\* )? ')'

Argument ::= EqualityExpr

 | InEqualityExpr

 | PathExpr

 | Number

 | Literal

Literal ::= '"' [^"]\* '"'

| "'" [^']\* "'"

Number ::= Digits ('.' Digits?)?

| '.' Digits

Digits ::= [0-9]+

FunctionExcluded ::= 'position()'

 | 'last()'

 | 'comment'

| 'processing-instruction'

| 'node'

FunctionName ::= Name - FunctionExcluded

Name     ::= NameStartChar (NameChar)\*

NameChar ::= NameStartChar | "-" | "." | [0-9]

NameStartChar ::= [A-Z] | "\_" | [a-z]

|  |
| --- |
| **End of modifications** |