

OASIS ebXML Registry

Proposal: XPATH Subset Syntax for Filter Query

Category: Improvements to existing specifications

Date: October 10, 2001

Author: Farrukh Najmi

Status of this Document

This document is a draft proposal whose purpose is to solicit additional input.

1 Abstract

The RS 1.0 specification defines a HasPathBranch element for Classification related queries as defined by the HasClassificationBranch. It is highly desirable to define a path pattern matching syntax that can be used in the HasPathBranch.

Currently the HasPathBranch is defined as follows in the latest version of filter query proposal:

```
<!ELEMENT HasPathBranch ( PathFilter | XpathNodeExpression | PathElementFilter+ )>
```

```
<!ELEMENT XpathNodeExpression ( TO BE DETERMINED )>
```

This document proposes a syntax for a proper sub-set of XPATH syntax that can be used as a pattern matching syntax for the CDATA for the `XpathNodeExpression` element.

2 Motivation

The following motivations drive this proposal:

1. Allow simple and intuitive syntax for doing path based filtering for classification related queries.

2.1 Assumptions

The following assumptions are made in this proposal:

28 Issues dealing with multiple co-operating registries are not considered. These
 29 issues are deferred to the Inter Registry Cooperation (IRC) team.

30 **3 Changes To FilterQuery Proposal**

31 The following changes need to be made to latest filter query proposal.

32 Replace line 222 with the following:

33 `<!ELEMENT XPathNodeExpression (#PCDATA)>`

34 Add 3.1, 3.2 and 3.3 where appropriate in section 8.2.

35 **3.1 XPathNodeExpression**

36 The `XpathNodeExpression` element must include a `pathFilter` string in its content as
 37 defined by following XML Schema definition:

```
38
39 <element name = "XPathNodeExpression">
40     <complexType>
41         <attribute name = "pathFilter" use = "required" type = "string"/>
42     </complexType>
43 </element>
```

44

45 The path filter expression is used to match classification nodes during the
 46 classification related filter query involving either internal or external classification.

47 The path filter expressions are based on a very small and proper sub-set of
 48 location path syntax of XPATH.

49 The path filter expression syntax includes support for matching multiple nodes by
 50 using wild card syntax as follows:

- 51 ○ Use of '*' as a wildcard in place of any path element in the `pathFilter`
- 52 ○ Use of '/' syntax to denote any descendent of a node in the `pathFilter`

53 It is defined by the following BNF grammar:

54

```
55 pathFilter ::= '/' schemeld nodePath
56 nodePath  ::= slashes nodeCode
57           | slashes '*'
58           | slashes nodeCode ( nodePath )?
59 Slashes   ::= '/' | '/'
```

60

61 In the above grammar, schemeld is the id attribute of the ClassificationScheme
 62 instance. In the above grammar nodeCode is of type string as defined by
 63 <http://www.w3.org/TR/xmlschema-2/#string> .

64 **3.2 Use of Path Filter Expressions In ClassificationNodeQuery**

65 The semantic rules for the ClassificationNodeFilter element should be extended
 66 to allow the use of getPath method as a filter that is based on the EQUAL clause.
 67 The pattern specified for matching the EQUAL clause is a PATH Filter
 68 expression.

69 This is illustrated in the following example which matches all second level nodes
 70 in ClassificationScheme with id 'Geography-id' and with code 'Japan':

```
71 <ClassificationNodeQuery>
72     <ClassificationNodeFilter>
73         getPath EQUAL "/Geography-id/*/Japan"
74     </ClassificationNodeFilter>
75 </ClassificationNodeQuery>
```

76

77 **3.3 Use Cases and Examples of Path Filter Expressions**

78 The following table lists various use cases and examples using the sample
 79 Geography scheme below:

```
80
81 <ClassificationScheme id='Geography-id' name="Geography"/>
82
83 <ClassificationNode id="NorthAmerica-id" parent="Geography-id" code="NorthAmerica" />
84 <ClassificationNode id="UnitedStates-id" parent="NorthAmerica-id" code="UnitedStates" />
85
86 <ClassificationNode id="Asia-id" parent="Geography-id" code="Asia" />
87 <ClassificationNode id="Japan-id" parent="Asia-id" code="Japan" />
88 <ClassificationNode id="Tokyo-id" parent="Japan-id" code="Tokyo" />
```

89

Use Case	PATH Expression	Description
Match all nodes in first level that have a specified value	/Geography-id/NorthAmerica	Find all first level nodes whose code is 'NorthAmerica'
Find all children of first level node	/Geography-id/NorthAmerica/*	Match all nodes whose first level path element has code

whose code is "NorthAmerica"		"NorthAmerica"
Match all nodes that have a specified value regardless of level	/ Geography-id//Japan	Find all nodes with code "Japan"
Match all nodes in the second level that have a specified value	/Geography-id/*/Japan	Find all second level nodes with code 'Japan'
Match all nodes in the 3rd level that have a specified value	/ Geography-id/*/*/Tokyo	Find all third level nodes with code 'Tokyo'

90

91 **4 Examples of Usage in Filter Query**

92 To illustrate the simplifying benefits of path filter expressions consider the
 93 following use case. Lets say we want to find ALL ClassificationNodes in the
 94 subtree beneath the given ClassificationNode.

95 First let us consider how it could be done without using path expressions based
 96 on the latest filter query proposal. The following example is from line 667-680 of
 97 latest filter query proposal.

98

```

99 <ClassificationNodeQuery>
100   <FromSchemeBranch>
101     <ClassificationSchemeFilter> id EQUAL "urn:some:known:scheme"
102   </ClassificationSchemeFilter>
103   </FromSchemeBranch>
104
105   <HasParentBranch>
106     <HasPathBranch>
107       <PathFilter>path STARTSWITH "KnownPathOfGivenNode" </PathFilter>
108     </HasPathBranch>
  
```

109 </HasParentBranch>

110 </ClassificationNodeQuery>

111

112 The following path filter based Query would replace the preceding one:

113

114 <ClassificationNodeQuery>

115 <ClassificationNodeFilter>

116 getPath EQUAL "/urn:some:known:scheme/< KnownPathOfGivenNode>/*"

117 </ClassificationNodeFilter>

118 </ClassificationNodeQuery>