OASIS ebXML Registry

Proposal: XPATH Subset Syntax for Filter Query

Category: Improvements to existing specifications

Date: October 5, 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:

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

3 Changes To FilterQuery Proposal

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

Replace line 222 with the following:

<!ELEMENT XpathNodeExpression (#PCDATA)>

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

3.1 XpathNodeExpression

The XpathNodeExpression element must include a path filter expression in its content. The path filter expression is used to match classification nodes during the classification related filter query involving either internal or external classification.

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

The path expression syntax includes:

- Use of '*' as a wildcard in place of any path element
- Use of '//' syntax to denote any descendent of a node

It is defined by the following BNF grammar:

```
pathFilter      ::= '/' schemeId nodePath
nodePath        ::= slashes nodeCode
                |   slashes '*'
                |   slashes nodeCode ( nodePath )?
Slashes         ::= '/' | '//'

nodeCode        ::= ID

ID              ::= LETTER  ( "_" | DIGIT | LETTER )*
LETTER          ::= ["A"-"Z", "a"-"z"]
DIGIT           ::= ["0"-"9"]
```

In the above grammer, schemeId is the id attribute of the ClassificationScheme instance.

3.2 Use of Path Filter Expressions In ClassificationNodeQuery

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

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

```
<ClassificationNodeQuery>
    <ClassificationNodeFilter>
        getPath EQUAL "/Geography-id/*/Japan"
    </ClassificationNodeFilter>
</ClassificationNodeQuery>
```

3.3 Use Cases and Examples of Path Filter Expressions

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

```
<ClassificationScheme id='Geography-id' name="Geography”/>

<ClassificationNode id="NorthAmerica-id" parent="Geography" code=NorthAmerica" />
<ClassificationNode id="UnitedStates-id" parent="NorthAmerica" code="UnitedStates" />

<ClassificationNode id="Asia-id" parent="Geography" code="Asia" />
<ClassificationNode id="Japan-id" parent="Asia" code="Japan" />
<ClassificationNode id="Tokyo-id" parent="Japan" code="Tokyo" />
```

Use Case	PATH Expression	Description
Match all nodes in first level that have a specified value	/Geography-id/NorthAmerica	Find all first level nodes with code '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'

4 Examples of Usage in Filter Query

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

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

```
<ClassificationNodeQuery>
   <FromSchemeBranch>
      <ClassificationSchemeFilter> id EQUAL "urn:some:known:scheme"
</ClassificationSchemeFilter>
   </FromSchemeBranch>

   <HasParentBranch>
      <HasPathBranch>
         <PathFilter>path STARTSWITH "KnownPathOfGivenNode" </PathFilter>
      </HasPathBranch>
   </HasParentBranch>
</ClassificationNodeQuery>
```

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

```
<ClassificationNodeQuery>
   <ClassificationNodeFilter>
      getPath EQUAL "/urn:some:known:scheme/*"
   </ClassificationNodeFilter>
</ClassificationNodeQuery>
```