XPATH IN NETCONF AND YANG

tail-f Tail-f is now part of Cisco. CISCO

Table of Contents

1. Introduction

XPath is a powerful tool used by NETCONF and YANG. This application note will help you to understand and utilize this advanced feature of NETCONF and YANG. This application note gives a brief introduction to XPath, then describes how XPath is used in NETCONF and YANG, and finishes with a discussion of XPath in ConfD.

The XPath 1.0 standard was defined by the W3C in 1999. It is a language which is used to address the parts of an XML document and was originally design to be used by XML Transformations. XPath gets its name from its use of path notation for navigating through the hierarchical structure of an XML document.

Since XML serves as the encoding format for NETCONF and a data model defined in YANG is represented in XML, it was natural for NETCONF and XML to utilize XPath.

2. XPath 1.0 Introduction

XML Path Language, or XPath 1.0, is a W3C recommendation first introduced in 1999. It is a language that is used to address and match parts of an XML document.

XPath sees the XML document as a tree containing different kinds of nodes. The types of nodes can be root, element, text, attribute, namespace, processing instruction, and comment nodes. The types of nodes that we will be working with mainly in the context of NETCONF and YANG are element nodes. The root node is the root of the tree. An order is defined on all nodes except for attribute and namespace nodes.

XPath expressions are the primary syntactic construct in XPath. XPath expressions, when evaluated, either yield a node-set, a Boolean, a number, or a string. When evaluating an XPath expression, this is done within a context. The use of this notion of context will be explained when discussing XPath usage in NETCONF and YANG.

XPath uses path expressions to select nodes in an XML document, and this expression can be used to return a collection of nodes, rather than a unique node. For example, the path expression /sys/interfaces/interface may return a collection of interface nodes, each representing a specific interface.

XPath has a number of built-in functions. All of the built-in functions are supported. Commonly used functions include the string functions starts-with() and contains(), the Boolean functions, and the node-set functions, such as count().

Some examples of XPath expressions include the following:

- The concat(“hello”, “ “, “world)
- /sys/host (a path expression)
- count(/sys/host)

Now, we will illustrate the use of XPath by discussing the various ways that XPath is used in NETCONF and YANG

3. The Use of XPath in NETCONF

The use of XPath in NETCONF is governed by the optional XPath capability as defined in RFC 4741 (NETCONF 1.0) and RFC 6241 (NETCONF 1.1). Making XPath an optional capability allows for the implementation of simple NETCONF servers, but the presence of the XPath capability does give the NETCONF client a powerful tool.

The presence of the XPath capability indicates that the NETCONF server supports the use of XPath expressions in the <filter> element of <get> and <get-config> operations. Note that the XPath expression used in the <filter> element must return a node-set.

An example, where we want to retrieve the set of users where the name is ‘Fred’, we could send the following request, with the entries in the return RPC.

```
<rpc message-id=”101”
     xmlns=”urn:ietf:params:xml:ns:netconf:base:1.0”>
   <get-config>
     <source>
       <running/>
     </source>
     <!-- get the user named fred -->
     <filter xmlns:t=”http://example.com/schema/1.2/config”
        type=”xpath”
        select=”/t:top/t:users/t:user[t:name=’fred’]”/>
    </get-config>
 </rpc>
```

```
<rpc-reply message-id=”101”
    xmlns=”urn:ietf:params:xml:ns:netconf:base:1.0”>
   <data>
      <top xmlns=”http://example.com/schema/1.2/config”>
        <users>
          <user>
            <name>fred</name>
            <company-info>
              <id>2</id>
            </company-info>
          </user>
        </users>
      </top>
    </data>
 </rpc-reply>
```

4. The Use of XPath in YANG

YANG 1.1, defined in RFC 7950, saw the introduction of new XPath functions, so we will first cover the use of XPath that was introduced in YANG 1.0, RFC 6020, before discussing YANG 1.1.

YANG 1.0

YANG uses XPath as a way of specifying references between nodes, which we will see in YANG leafref statements, and dependencies between nodes, which we will see in YANG must and when statements.

Note that YANG does not mandate the use of an XPath interpreter and leaves it up to the implementer how enforcement of XPath expressions is accomplished

Must Statements

A YANG must statement is used to declare a constraint on valid data and is optional. When the must statement appears, it takes as an argument a valid XPath expression. The XPath context node, which can be used for example in relative paths, is the node on which the must statement appears.

The result of the XPath expression used in the must statement is converted to a Boolean using standard XPath rules.

In the following example, we want to ensure that the MTU size is set appropriately for interfaces of type Ethernet and ATM, so we add the appropriate must statements. One thing to note is that both must expressions are evaluated, so care must be taken to make sure both evaluate to true in valid cases.

```
container interface {
   leaf ifType {
     type enumeration {
       enum ethernet;
       enum atm;
     }
   }
   leaf ifMTU {
     type uint32;
   }
   must "ifType != 'ethernet' or " +
        "(ifType = 'ethernet' and ifMTU = 1500)" {
        error-message "An ethernet MTU must be 1500";
   }
   must "ifType != 'atm' or " +
        "(ifType = 'atm' and ifMTU <= 17966 and ifMTU >= 64)" {
        error-message "An atm MTU must be 64 .. 17966";
   }
 }
```

When Statements

Use of the when statement makes the data definition statement it is attached to conditional.

The argument to this statement is an XPath expression which formally expresses the condition. If the expression evaluates to true for a particular instance, then the data definition is valid. Otherwise, it is not.

Let's look at an example. Suppose we want to add an additional leaf to contain the uid in our list of users, if the class node is not 'wheel':

```
augment /system/login/user {
  when "class != 'wheel'";
  leaf uid {
    type uint16 {
      range "1000 .. 30000";
    }
  }
}
```

Path Statements

When you use the type leafref to define a node, then, as a substatement to the type, you must use a path statement. The XPath argument to the path statement must refer to a leaf or leaf-list node in the tree.

As an example, in your YANG model, you configure the management interface, but you want the management interface to be one of the existing interfaces. You can use the type leafref as the type of the management interface and have the path statement refer to the name of some interface in the list:

```
list interface {
  key “name”;
  leaf name {
    type string;
  }
}
leaf mgmt-interface {
  type leafref {
    path “../interface/name”;
  }
}
```

YANG 1.1

In YANG 1.1, based on experience gained with using YANG 1.0, several new XPath functions were added as a convenience to the YANG model writer.

YANG 1.0 had already defined the XPath function current(), which returns a node set containing the context node. YANG 1.1 adds the function for strings call re-match(), for the YANG type identityref the functions derived-from() and derived-from-or-self(), for the YANG type enumeration the function enum-value(), for the YANG type bits the function bit-is-set(), and for leafref and instance-identifier the function deref(). In this document, we’ll just describe the deref() function and invite you to read the RFC for details on the other functions.

One difficulty with using leafref, for example, is that when you have multiple leafs which must reference the same entry, sometimes writing the XPath expression can be cumbersome and error prone.

As an example of this, suppose you have a list of servers, with the ip and port being leafs in the list. If you have a particular configuration entry which selects one of those list entries, then you must write the path statements as follows:

```
leaf my-ip {
  type leafref {
    path “/server/ip”;
  }
}
leaf my-port {
  type leafref {
    path “/server[ip = current()/../my-ip]/port”;
  }
}
```

The path statement for my-port must first select the node-set which has the same IP address as my-ip and then select from among the possible ports that have that IP address.

To simplify the path statement, we can use the deref() function:

```
leaf my-ip {
  type leafref {
    path "/server/ip";
  }
}
leaf my-port {
  type leafref {
    path "deref(../my-ip)/../port";
  }
}
```

5. XPath and ConfD

For NETCONF, XPath is an optional capability, but it is implemented and supported in ConfD, so the ConfD NETCONF server can advertise XPath as a capability. This means that the NETCONF client is able to use XPath as the filter in NETCONF <get> and <get-config> RPCs.

ConfD has its own XPath 1.0 implementation. So, it has full support for must, when, and path statements in YANG data models and automatically processes and enforces any XPath expressions used in a YANg data model.

When developing YANG must and when statements, it can be difficult to verify that the statement works as intended. To assist ConfD developers with this task, ConfD provides a CLI command, xpath, which can be used to test XPath expressions.

The way to enable this command is to set devtools to true in a CLI session, by entering the command devtools true. Once that is done, the xpath command is available in configure mode. The syntax of the command is as follows:

```
xpath [ctx <path>] (eval | must | when) <expression>
```

You can optionally specify the XPath context, otherwise the context is the current sub-mode. You can then have your XPath expression simply evaluated, treated as a YANG must, or when statement.

Further information on the xpath command can be found in the section "Configure mode commands" in the chapter "The CLI agent" in the ConfD User's Guide.

6. Conclusion

XPath provides some powerful constructs for filtering data when fetching information via NETCONF. It also provides useful tools for YANG data model writers for specifying constraints in YANG data models. ConfD includes a full XPath implementation which allows users to take advantage of XPath as it is used in the NETCONF and YANG RFCs. networks. YANG provides the specification of the data available, as well as a structure for how the data is organized. NETCONF provides the methods to access the data and manipulate it.

7. Additional Resources

NW3Schools XPath tutorial – Basic tutorial:
http://www.w3schools.com/xml/xpath_intro.asp

Writing XPath expressions – More complex:
http://www.stylusstudio.com/docs/v2009r2/d_xpath.html

To learn more about ConfD, visit http://www.tail-f.com

Tail-f is now part of Cisco.

www.tail-f.com
info@tail-f.com

Corporate Headquarters

Sveavagen 25
111 34 Stockholm
Sweden
+46 8 21 37 40