Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 12, 2013

A. Bierman
YumaWorks

September 8, 2012

The NETCONF <get2> Operation
draft-bierman-netconf-get2-00

Abstract

This document describes NETCONF protocol enhancements to improve data retrieval capabilities.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 12, 2013.

Copyright Notice

Table of Contents

1. Introduction

There is a need for standard mechanisms to allow NETCONF [RFC6241] application designers to retrieve data from NETCONF servers more efficiently.

1.1. Problem Statement

This document attempts to address the following problems with NETCONF data retrieval mechanisms.

1.1.1. Too Much Data Returned

The NETCONF <get> operation allows a client to retrieve data from the server but it returns all data, including configuration datastore nodes. The <get-config> operation already returns all configuration datastore nodes.

It was originally thought that <get> should return all nodes so the client would not have to correlate configuration and non-configuration data nodes, since they would be mixed together in the reply.

Operational experience has shown that the <get> operation without reasonable filters to reduce the returned data can significantly degrade device performance and return enormous XML instance documents in the <rpc-reply>.

1.1.2. No Last-Modified Indication or Time Filtering

The NETCONF protocol has no standard mechanisms to indicate to a client when a datastore was last modified, or to allow a client to retrieve data only if it has been modified since a specified time. This makes polling applications very inefficient because they will regularly burden the server and the network and themselves with retrieval and processing requests for data that has not changed.

1.1.3. No Simple Instance Discovery Mechanism

Sometimes the client application wants to discover what data exists on the server, particularly list entries. There is a need for a simple mechanism to retrieve just the key leaf nodes within a subtree.

The NETCONF subtree filtering mechanism does provide a very complex way for the client to request just key leafs for specific list entries. A simpler mechanism is needed which will allow the client to discover the list instances present.

1.1.4. No Subtree Depth Control

NETCONF filters allow the client to select specific sub-trees within the conceptual datastore on the server. However, sometimes the client does not really need the entire subtree, which may contain many nested list entries, and be very large.

There is sometimes a need to limit the depth of the sub-trees retrieved from the server. A consistent and simple algorithm for determining what data nodes start a new level is needed.

1.1.5. Content Filter Specification is not Extensible

The NETCONF <get> and <get-config> operations use a hard-coded content filtering mechanism. They use a 'type' XML attribute to indicate which of two filter specification types they support, and a 'select' XML attribute if the :xpath capability is supported and an XPath [XPATH] expression filter specification is provided.

This design does not allow additional content filter specification types to be supported by an implementation. It does not allow the standard to be easily extended in a modular fashion.

In addition, this design does not allow YANG statements to be used to properly describe the protocol operation. The special 'get-filter-element-attributes' YANG extension in the ietf-netconf module is not extensible, and it does not really count as proper YANG, since this extension is outside the YANG language definition.

1.1.6. Subtree Filtering is Mandatory

There is a mandatory-to-implement subtree filter mechanism in NETCONF. There are server environments where subtree filtering cannot be supported or is not needed for the specific device or supported management application(s).

A NETCONF server SHOULD support subtree filtering. An implementation MAY omit subtree filtering support if it is not applicable or not feasible on the device.

1.2. Solution

This document defines a new NETCONF protocol operation called <get2> to address the deficiencies described in the previous section. It can be implemented existing NETCONF servers without requiring a change in the protocol version.

1.3. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119].

1.3.1. NETCONF

The following terms are defined in [RFC6241]:

- candidate configuration datastore
- client
- configuration data
- datastore
- configuration datastore
- protocol operation

- running configuration datastore
- server
- startup configuration datastore

1.3.2. YANG

The following terms are defined in [RFC6020]:

- anyxml
- container
- data node
- key leaf
- leaf
- leaf-list
- list

- presence container (or P-container)
- non-presence container (or NP-container)

1.3.3. Terms

The following terms are defined:

- operational datastore: the collection of all conceptual YANG data nodes which represent non-configuration data. This conceptual datastore also includes ancestor container and list nodes for any nested non-configuration nodes, as well as list keys for any list nodes in this datastore.
- depth filter: A mechanism implemented within the NETCONF server to

allow a client to retrieve only a limited number of levels within the a subtree, instead of retrieving the entire subtree.

- time filter: A mechanism implemented within the NETCONF server to allow a client to retrieve only data that has been modified since a specified data and time.

2. <get2> Operation

The <get2> operation is defined with a YANG 'rpc' statement. A specific datastore is selected for the source of the retrieval operation. Several different types of filters are provided. Filters are combined in a conceptual "logical-AND" operation, and are optional to use by the client. Not all filtering mechanisms are mandatory-to-implement for the server.

The <get2> protocol operation contains the following input parameters:

- source: A container indicating the conceptual datastore for the retrieval request.
- filter-spec: A choice indicating the content filter specification for the retrieval request.
- keys-only: A leaf indicating that only the key leafs, combined with other filtering criteria, should be returned.
- if-modified-since: A leaf indicating the time filter specification for the retrieval request, according to the procedures in Section 2.2.
- depth: A leaf indicating the subtree depth level for the retrieval request, according to the procedures in Section 2.1.
- with-defaults: A leaf indicating the type of defaults handling requested, according to procedures in [RFC6243].
- with-timestamps: A leaf indicating that 'last-modified' XML attributes are requested, according to procedure in Section 3.

2.1. Depth Filters

A depth filter indicates how many subtree levels should be returned in the <rpc-reply>. This filter is specified with the 'depth' input parameter for the <get2> protocol operation. The default '0' indicates that all levels from the requested subtrees should be returned.

A new level is started for each YANG P-container or list within the descendant nodes of the requested subtree.

For purposes of counting levels, NP-containers are transparent. The contents of an NP container are considered to be at the same level as the NP-container itself.

YANG data nodes of type 'leaf', 'leaf-list', and 'anyxml' are

considered to be at the same level as their parent container or list.

If no content filters are provided, then level 1 is considered to include all top-level data nodes within the source datastore. Otherwise only the levels in selected subtrees will be considered, and not any additional top-level data nodes.

2.2. Time Filters

A time filter indicates that only data which has been modified since the indicated date and time should be included in the reply.

If this feature is supported, then the server will maintain a last-modified timestamp for the source datastore. It MAY support additional nested timestamps for data nodes within the datastore.

When a request containing the 'if-modified-since' parameter is received, the server will compare that timestamp to the last-modified timestamp for the source datastore. If it is greater than the specified value then data may be returned (depending on other filters). If the datastore timestamp value is less than or equal to the specified value, then an empty <data> element will be returned in the <rpc-reply>.

If the server maintains 'last-modified' timestamps for any data nodes within the source datastore then the same type of comparison will be done for the data node to determine if it should be included in the response. If no 'last-modified' timestamp is maintained for a data node, then the server will use the 'last-modified' timestamp for its nearest ancestor, or for the datastore itself if there are none.

3. XSD for the 'last-modified' Attribute

The following XML Schema document [XSD] defines the 'last-modified' attribute, described within this document. This XSD is only relevant if the server supports the 'timestamps' YANG feature within the 'ietf-netconf-get2' YANG module.

The 'last-modified' attribute uses the XSD data type 'dateTime', in accordance with Section 3.2.7.1 of XML Schema Part 2: Datatypes. This is equivalent to the YANG data type 'date-and-time'.

```
<CODE BEGINS> file="last-modified.xsd"

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:ietf:params:xml:ns:netconf:default:1.0"
  targetNamespace="urn:ietf:params:xml:ns:netconf:last-modified:1.0"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  xml:lang="en">

 <xs:annotation>
   <xs:documentation>
     This schema defines the syntax for the 'last-modified' attribute
     described within this document.
   </xs:documentation>
 </xs:annotation>

 <!--
   last-modified attribute
   -->
 <xs:attribute name="last-modified" type="xs:dateTime">
   <xs:annotation>
     <xs:documentation>
       This attribute indicates the date and time when
       a modification was last detected by the server
       for the datastore or data node corresponding to
       the XML element containing this attribute.
     </xs:documentation>
   </xs:annotation>
 </xs:attribute>
</xs:schema>

<CODE ENDS>
```

4. YANG Module

This module imports the 'with-defaults-parameters' grouping from [RFC6243].

Several YANG features are imported from [RFC6241].

Some data types are imported from [RFC6021].

RFC Ed.: update the date below with the date of RFC publication and remove this note.

<CODE BEGINS> file "ietf-netconf-get2@2012-09-08.yang"

```
module ietf-netconf-get2 {

  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-get2";
  prefix get2;

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-netconf {
    prefix nc;
  }

  import ietf-netconf-with-defaults {
    prefix ncwd;
  }

  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

  contact
```

```
       "WG Web:   <http://tools.ietf.org/wg/netconf/>
        WG List:  <mailto:netconf@ietf.org>

        WG Chair: Mehmet Ersue
                  <mailto:mehmet.ersue@nsn.com>

        WG Chair: Bert Wijnen
                  <mailto:bertietf@bwijnen.net>
```



```
        Editor:   Andy Bierman
                  <mailto:andy@yumaworks.com>";

     description
       "This module contains a collection of YANG definitions for the
        retrieval of information from a NETCONF server.

        Copyright (c) 2012 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

     // RFC Ed.: replace XXXX with actual RFC number and remove this
     // note.

     // RFC Ed.: remove this note
     // Note: extracted from draft-bierman-netconf-get2-00.txt

     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.
     revision "2012-09-08" {
       description
         "Initial revision.";
       reference
```

```
      "RFC XXXX: The NETCONF <get2> Operation";
  }

  /* Features */

  feature timestamps {
    description
      "This feature indicates that the server implements
       the <get2> operations parameters which require
       last modification timestamps to be maintained by
       the server.

       If this feature is advertised then one global
       'last-modified' timestamp for the entire
       running datastore MUST be supported.
```



```
       The server MAY support additional timestamps
       for additional datastores and data nodes
       within a datastore.  The 'with-timestamps'
       parameter can be used to identify
       which data nodes support a last-modified-time
       timestamp.";
  }

  feature with-defaults {
    description
      "This feature indicates that the server supports the
       'with-defaults' parameter for the <get2> operation.
       A NETCONF server SHOULD support this feature.";
    reference
      "RFC 6243: With-defaults Capability for NETCONF";
  }

  feature subtree-filter {
    description
      "This feature indicates that the server supports the
       subtree filtering mechanism for selecting portions
       of the selected source datastore.
       A NETCONF server SHOULD support this feature.";
    reference
      "RFC 6241: NETCONF Protocol, section 6.";
```

```
}

/* Protocol Operations */

rpc get2 {
  description
    "Retrieve NETCONF datastore information";
  input {
    container source {
      choice datastore-source {
        default running;
        description
          "The configuration source for the retrieval operation.
           The running configuration is the default choice if
           this parameter is not present.";
        leaf candidate {
          if-feature nc:candidate;
          type empty;
          description
            "The candidate configuration datastore is the
             retrieval source.";
        }
        leaf running {
```



```
          type empty;
          description
            "The running configuration datastore is the
             retrieval source.";
        }
        leaf operational {
          type empty;
          description
            "The collection of non-configuration
             data nodes supported by the server is the
             retrieval source.";
        }
        leaf startup {
          if-feature nc:startup;
          type empty;
          description
            "The startup configuration datastore is the
             retrieval source.";
```

```
        }
        leaf url {
          if-feature nc:url;
          type inet:uri;
          description
            "The URL-based configuration is the
             retrieval source.";
        }
      }
    }

    choice filter-spec {
      anyxml filter {
        if-feature subtree-filter;
        description
          "This parameter identifies the portions of the
           target datastore to retrieve.";
        reference "RFC 6241, Section 6.";
      }
      leaf select {
        if-feature nc:xpath;
        type yang:xpath1.0;
        description
          "This parameter contains an XPath expression
           identifying the portions of the target
           datastore to retrieve.";
      }
    }

    leaf keys-only {
```



```
      type empty;
      description
        "This parameter selects only data nodes which
         are key leaf nodes.  Parent container and
         list nodes are also returned, but no other leafs,
         or any leaf-lists will be included in the reply.";
    }

    leaf if-modified-since {
      if-feature timestamps;
      type yang:date-and-time;
```

```
        description
          "This parameter selects only data nodes which
           have been modified since the specified time.";
      }

      leaf depth {
        type uint32;
        default 0;
        description
          "This parameter selects how many conceptual
           sub-tree levels should be returned in the
           <rpc-reply>.

           If this parameter is equal to '0', then entire
           subtrees will be returned.

           If this parameter is greater than '0', then
           only the specified number of subtree levels will
           be returned.";
        reference "RFC XXXX, section 2.1.";
      }

      uses ncwd:with-defaults-parameters {
        if-feature with-defaults;
        description
          "This parameter controls the retrieval of
           default values.";
         reference
           "RFC 6243: With-defaults Capability for NETCONF";
      }

      leaf with-timestamps {
        if-feature timestamps;
        type empty;
        description
          "This parameter will cause the server to return
           XML attributes identifying the last modification
```



```
           time within one or more elements within the
           <rpc-reply>.";
        reference "RFC XXXX, sections 2.2 and 3.";
      }
```

```
      }

      output {
        anyxml data {
          description
            "Copy of the requested datastore subset which
             matched the filter criteria (if any).
             An empty data container indicates that the
             request did not produce any results.";
        }
      }
    }

  }

  <CODE ENDS>
```

5. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-get2

Registrant Contact: The NETCONF WG of the IETF.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

```
name:         ietf-netconf-get2
namespace:    urn:ietf:params:xml:ns:yang:ietf-netconf-get2
prefix:       get2
reference:    RFC XXXX
```

6. Security Considerations

This document does not introduce any new security concerns in addition to those specified in [RFC6241], section 9.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6243] Bierman, A. and B. Lengyel, "With-defaults Capability for NETCONF", RFC 6243, June 2011.

[XPATH] Clark, J. and S. DeRose, "XML Path Language (XPath) Version 1.0", World Wide Web Consortium Recommendation REC-xpath-19991116, November 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116>.

[XSD] Malhotra, A. and P. Biron, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

7.2. Informative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
           January 2004.

Appendix A. Examples

A.1. YANG Module Used in Examples

```
   module example-get2 {

     namespace "http://example.com/ns/example-get2";
     prefix exget2;

     revision 2012-09-08;

     container forests {
       list forest {
         key name;

         leaf name {
           type string;
         }

         leaf tree-count {
           config false;
           type uint32;
         }

         container trees {
           list tree {
```

```
            key name;

            leaf name {
              type string;
            }
            leaf location {
              type string;
            }
            leaf height {
              config false;
              type decimal64 {
                fraction-digits 3;
              }
              units meters;
            }
          }  // list tree
        }  // container trees
      } // list forest
    }  // container forests
  }
```


A.2. YANG Data Used in Examples

The follow instances are assumed in the following examples.

```
  list forest: 'north':
    list tree: 'birch', 'ash', 'maple'

  list forest: 'south':
    list tree: 'banyan', 'palm'
```

A.3. Example: Operational Datastore

This example simply retrieves the 'forests' subtree data from the operational datastore.

```
  <rpc message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get2 xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-get2">
      <source>
```

```
        <operational />
      </source>
      <filter>
        <forests xmlns="http://example.com/ns/example-get2" />
      </filter>
    </get2>
  </rpc>

  <rpc-reply message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-get2">
      <forests xmlns="http://example.com/ns/example-get2">
        <forest>
          <name>north</name>
          <tree-count>3</tree-count>
          <trees>
            <tree>
              <name>birch</name>
              <height>41.013</height>
            </tree>
            <tree>
              <name>ash</name>
              <height>16.523</height>
            </tree>
            <tree>
              <name>maple</name>
              <height>51.204</height>
            </tree>
          </trees>
```



```
        </forest>
        <forest>
          <name>south</name>
          <tree-count>2</tree-count>
          <trees>
            <tree>
              <name>banyan</name>
              <height>91.433</height>
            </tree>
            <tree>
              <name>palm</name>
              <height>83.439</height>
```

```
            </tree>
          </trees>
        </forest>
      </forests>
    </data>
  </rpc-reply>
```

A.4. Example: If-Modified-Since Non-Empty Filter Retrieval

In this example, the running datastore was last modified at '2012-09-09T01:43:27Z' because the forest named 'north' was modified at this time.

The forest named 'north' was last modified after the specified 'if-modified-since' timestamp. The forest named 'south' was last modified before the specified 'if-modified-since' timestamp.

The server maintains a last-modified timestamp for the running datastore and the 'forest' list entries.

The client is also requesting that timestamps be returned for the nodes that have been modified. If any part of the 'forest' subtree is modified then this timestamp will be updated.


```
  <rpc message-id="102"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get2 xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-get2">
      <filter>
        <forests xmlns="http://example.com/ns/example-get2" />
```

```
        </filter>
        <if-modified-since>2012-09-09T01:43:27Z</if-modified-since>
        <with-timestamps />
      </get2>
    </rpc>

    <rpc-reply message-id="102"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
      xmlns:lm="urn:ietf:params:xml:ns:netconf:last-modified:1.0">
      <data
        xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-get2"
        lm:last-modified="2012-09-09T02:00:00Z">

        <forests xmlns="http://example.com/ns/example-get2">
          <forest lm:last-modified="2012-09-09T02:00:00Z">
            <name>north</name>
            <trees>
              <tree>
                <name>birch</name>
                <location>hillside</location>
              </tree>
              <tree>
                <name>ash</name>
                <location>southwest pasture</location>
              </tree>
              <tree>
                <name>maple</name>
                <location>east meadow</location>
              </tree>
            </trees>
          </forest>
      </data>
    </rpc-reply>
```

A.5. Example: If-Modified-Since Empty Filter Retrieval

In this example the client has changed the if-modified-since timestamp to a time in the future. No 'forest' list entry has been modified since this time so an empty data node is returned. Note that the last-modified timestamp is returned for the node representing the datastore, even though no data nodes have been modified since the specified time. This allows the client to easily retrieve the last-modified timestamp for the entire datastore.

```
<rpc message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get2 xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-get2">
    <filter>
      <forests xmlns="http://example.com/ns/example-get2" />
    </filter>
    <if-modified-since>2012-09-09T03:43:27Z</if-modified-since>
    <with-timestamps />
  </get2>
</rpc>

<rpc-reply message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:lm="urn:ietf:params:xml:ns:netconf:last-modified:1.0">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-get2"
    lm:last-modified="2012-09-09T02:00:00Z" />
</rpc-reply>
```

A.6. Example: Keys Only Filter Retrieval

This example retrieves the names-only from the 'forests' subtree in the running datastore. Only one subtree level is requested, instead of the default of all levels. The default source (running) is used. The default depth='0' is used to retrieve all subtree levels.

```
<rpc message-id="105"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get2 xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-get2">
    <filter>
      <forests xmlns="http://example.com/ns/example-get2" />
    </filter>
    <keys-only />
  </get2>
</rpc>

<rpc-reply message-id="105"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-get2">
    <forests xmlns="http://example.com/ns/example-get2">
      <forest>
        <name>north</name>
        <trees>
          <tree>
            <name>birch</name>
          </tree>
          <tree>
            <name>ash</name>
          </tree>
          <tree>
            <name>maple</name>
          </tree>
        </trees>
      </forest>
      <forest>
        <name>south</name>
        <trees>
          <tree>
            <name>banyan</name>
          </tree>
          <tree>
            <name>palm</name>
          </tree>
        </trees>
      </forest>
    </forests>
  </data>
</rpc-reply>
```

A.7. Example: Keys Only Filter Retrieval with Depth=1

This example retrieves the names-only from the 'forests' subtree in the running datastore. Only one subtree level is requested, instead of the default of all levels. The default source (running) is used.

The depth parameter is set to '1' to only retrieve the first layer, which includes the 'forests' NP container and the 'forest' list.

```
<rpc message-id="105"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get2 xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-get2">
    <filter>
      <forests xmlns="http://example.com/ns/example-get2" />
    </filter>
    <keys-only />
    <depth>1</depth>
  </get2>
</rpc>

<rpc-reply message-id="105"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-get2">
    <forests xmlns="http://example.com/ns/example-get2">
      <forest>
        <name>north</name>
      </forest>
      <forest>
        <name>south</name>
      </forest>
    </forests>
  </data>
</rpc-reply>
```

Author's Address

Andy Bierman
YumaWorks

Email: andy@yumaworks.com