NETCONF
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2021

C. Feng
Q. Ma
Huawei

October 20, 2020

With System Capability for NETCONF
draft-ma-netconf-with-system-00

Abstract

The NETCONF protocol [RFC6241] defines ways to read configuration and state data from a NETCONF server. In some cases, a client-configured data item refers to a non-existent system generated data item (e.g.,the auto-create interfaces ("eth1") is not yet present). In many situations, the system configured data item doesn't need to be know to the client and client-configured data item will automatically be removed from the operational state datastore and thus only appear in the intended datastore if client-configured data item doesn't exist. In other situations system configured data item needs to be known and overriden by the client. Not all server implementations treat the system configuration data in the same way. This document defines a capability-based extension to the NETCONF protocol that allows the NETCONF client to identify how system configuration are processed by the server, and also defines a new mechanism for client control of server processing of system configuration data.

Table of Contents

1. Introduction

The NETCONF protocol [RFC6241] defines ways to read configuration and state data from a NETCONF server.

In some cases, a client-configured data item refers to a nonexistent system generated data item (e.g.,the auto-create interfaces ("eth1") is not yet present).

- In many situations, the system configured data item doesn't need to be known to the client and client-configured data item will automatically be removed from the operational state datastore and thus only appear in the intended datastore if client-configured data item doesn't exists.

- In other situations system configured data item needs to be known and overriden by the client.

Not all server implementations treat the system configuration data in the same way.

This document defines a capablity-based extension to the NETCONF protocol that allows the NETCONF client to identify how system configuation are processed by the server, and also defines new mechanism for client control of server processing of system configuration data.

1.1. Terminology

This document assumes that the reader is familiar with the contents of [RFC6241], [RFC7950], [RFC8342], [RFC8407], and [RFC8525] and uses terminologies from those documents.

The following terms are defined in this document as follows:

System configuration: Configuration that is supplied by the device itself [RFC8342].

Logical resource dependent system configuration: When the device is powered on, the pre-provisioned configuration will be activated and provided, irrespective of physical resource present or not, sometimes the pre-provisioned configuration will be provided unconditionally(e.g., loop back interface activation), sometimes not, e.g., only provided when a special functionality is enabled.

Physical resource dependent system configuration: When the device is powered on and the physical resource is present(e.g., insert interface card), the system will automatically detect it and load pre-provisioned configuration; when the physical resource is not present(remove interface card), the system configuration will be automatically cleared.

System configuration datastore: A configuration datastore holding the complete system configuration of the device. This datastore is referred to as "<system>".

1.2. Requirements Language

The key words "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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. System Configuration Datastore

Following guidelines for defining Datastores in the appendix A of [RFC8342], this document introduces a new datastore resource named 'system' that represents the pre-provisioned configuration or physical resource dependent configuration.

- Name: "system"
- YANG modules: all
- YANG nodes: all "config true" data nodes
- Management operations: The content of the datastore is set by the server in an implementation dependent manner. The content can not be changed by management operations via NETCONF, RESTCONF,the CLI etc unless specialized, dedicated operations are provided. The datastore can be read using the standard NETCONF/RESTCONF protocol operations.
- Origin: This document does not define any new origin identity when it interacts with <operational> datastore. The system origin Metadata Annotation is used to indicate the origin of a data item.
- Protocols: RESTCONF, NETCONF and other management protocol.
- Defining YANG module: "ietf-netconf-with-system".

The datastore content is usually defined by the device vendor. It is static at most of time and MAY change e.g., depending on external factors like HW available or during device upgrade. On devices that support non-volatile storage, the contents of <system> configuration datastore need to be persist across reboots.

2.1. Life Cycle of the system configuration

When the device is powered on and the physical resource is inserted into the device, physical resource dependent system configuration will be automatically loaded into <system>;

When the physical resource is removed from the device, the physical resource dependent system configuration will be automatically removed from <system>;

When the Logical resource dependent system configuration is referenced by another configured data item, it will be automatically loaded into <system> if it doesn't exist in the <system>; The <system> will keep unchanged if the Logical resource dependent system configuration has already been existed in the <system>.

3. System Configuration Data Handling

The system configuration data handling behavior used by a server will impact NETCONF protocol operations in two ways:

- Data retrieval: A server is normally allowed to exclude data nodes which it considers to contain the system configuration data. The actual nodes omitted depends on the system configuration data handling behavior used by the server.

- Create and delete operations: The <edit-config> 'operation' attribute can be used to create and/or delete specific data nodes. These operations depend on whether the target node currently exists or not. The server's system configuration data handling behavior will determine whether the requested node currently exists in the configuration datastore or not.

4. System Configuration data handling Basic Modes

Not all server implementations treat system configuration data in the same way. Instead of forcing a single implementation strategy, this document allows a server to advertise a particular style of system configuration data handling, and the client can adjust accordingly.

NETCONF servers report system configuration data in different ways. This document specifies two standard defaults handling basic modes that a server implementor may choose from:

- report-all
- explicit

A server that uses the 'report-all' basic mode MUST automatically

- Update <running> with the system configuration, after the "system" configuration has been altered as a consequence of a plug and play operation or device powering on operation.
- The system configuration doesn't need to be explicitly set by the client first before the system configuration needs to be updated with client set configuration or referenced by client set configuration.

A server that uses the 'explicit' basic mode

MUST not update <running> with the system configuration,

The system configuration MUST be explicitly set by the client first before the system configuration needs to be updated with client set configuration or referenced by client set configuration.

4.1. Initialization During Reboot

On devices that support non-volatile storage, the contents of <system> (e.g.,logical resource dependent system configuration (conditional or unconditional) and physical resource dependent system configuration) will typically persist across reboots via that storage. At boot time, the device loads the saved system configuration into <running> together with saved startup configuration via 'merge' protocol operation. To save a new system configuration, data is copied to <system> via either implicit or explicit protocol operations.

4.2. 'report-all' <edit-config> Behavior

The server MUST consider every data node to exist, even those explicitly set by the server.

- A valid 'create' operation attribute for a data node that is loaded from <system> and explicitly set by the server MUST fail with a 'data-exists' error-tag;

- A valid 'delete' operation attribute for a data node that is loaded from <system> and explicitly set by the server MUST succeed. The deleted system configuration MUST be reloaded into <running> immediately if the system configuration is still present in the <system>;

- A valid 'merge' operation attribute for a data node that is loaded from <system> and explicitly set by the server MUST succeed.

4.3. 'explicit' <edit-config> Behavior

The server considers any data node that is explicitly set data to exist.

- A valid 'create' operation attribute for a data node that is explicitly set by the server MUST succeed since the system configuration data is kept in the <running> configuration datastore.

- A valid 'merge' operation attribute for a data node that is explicitly set by the server MUST succeed even though the name of data node explicitly set by the server is same as name of data node explicitly set by the client.

- A valid 'delete' operation attribute for a data node that is explicitly set by the client MUST succeed even though the name of data node explicitly set by the server is same as name of data node explicitly set by the client. A valid 'delete' operation attribute for a data node that is not explicitly set by the client MUST fail if system configuration is not loaded into <running>.

5. Retrieval of System Configuration Data

When data is retrieved from a server using the 'explicit' basic mode, and the <with-system> parameter is not present, data nodes MUST be reported if explicitly modified by the client.

If the 'explicit' basic mode is used by the server and the <with-system> parameter supported by the server is set to a value equal to 'explicit', data nodes MUST be reported if explicitly modified by the client.

When data is retrieved from a server using the 'report-all' basic mode, and the <with-system> parameter is not present, all data nodes MUST be reported without data nodes considered to be system configuration data by the server.

If the 'report-all' basic mode is used by the server and the <with-system> parameter supported by the server is set to a value equal to 'report-all', all data nodes MUST be reported, including any data nodes considered to be system configuration data by the server.

If the 'report-all' basic mode is used by the server and the <with-system> parameter supported by the server is set to a value equal to 'report-all-tagged', all data nodes MUST be reported, including any data nodes considered to be system configuration data by the server. Explicitly set data by the server will be tagged with <system> if the system configuration is applied.

When data is retrieved from a server using the <with-system> parameter with a value equal to 'trim' , data nodes MUST be reported if considered to be system configuration data by the server. Data node MUST NOT be reported if explicitly modified by the client.

6. With System Capability

6.1. Overview

The :with-system capability indicates which system-data-handling basic mode is supported by the server. These retrieval modes allow a NETCONF client to control whether system configuration data is returned by the server. Sending of system configuration data is controlled for each individual operation separately.

A NETCONF server implementing the :with-system capability:

- MUST indicate its basic mode behavior by including the 'basic-mode' parameter in the capability URI;
- MUST support the YANG module defined in Section 7 for the system configuration data handling mode indicated by the 'basic-mode' parameter.
- SHOULD support the YANG module in Section 7 for the system configuration data handling mode identified by the 'report-all' or 'report-all-tagged' enumeration value.
- If the 'report-all-tagged' system data handling mode is supported, then the 'origin' metadata attribute MUST be supported.
- MAY support the YANG module in Section 7 for additional system data handling modes.

6.2. Capability Identifier

urn:ietf:params:netconf:capability:with-system:1.0

The identifier MUST have a parameter: "basic-mode". This indicates how the server will treat system configuration data, as defined in Section 4. The allowed values of this parameter are 'report-all', and 'explicit', as defined in Section 4.

The identifier MAY have another parameter: "also-supported". This parameter indicates which additional enumeration values (besides the basic-mode enumeration), the server will accept for the <with-system> parameter in Section 4. The value of the parameter is a comma separated list of one or more modes that are supported beside the mode indicated in the 'basic-mode' parameter. Possible modes are 'report-all', 'report-all-tagged','trim' and 'explicit', as defined in Section 4.

urn:ietf:params:netconf:capability:with-system:1.0?basic-mode=explicit&also-supported=report-all,report-all-tagged

6.3. Modifications to Existing Operations

6.3.1. <get> and <get-config> Operations

A new <with-system> XML element is added to the input for the <get>, <get-config> and <copy-config> operations. If the <with-system> element is present, it controls the reporting of system configuration data. The server MUST return system configuration data in the NETCONF <rpc-reply> messages according to the value of this element, if the server supports the specified retrieval mode.

This parameter only controls these specified retrieval operations, and does not impact any other operations or the non-volatile storage of configuration data.

The <with-system> element is defined in the XML namespace for the ietf-netconf-with-system.yang module in Section 7, not the XML namespace for the <get>, <get-config> and <copy-config> operations.

If the <with-system> element is not present, the server MUST follow its basic mode behavior as indicated by the :with-system capability identifier's 'basic-mode' parameter, defined in Section 6.2.

The <get> and <get-config> operations support a separate filtering mechanism, using the <filter> parameter. The system configuration data filtering is conceptually done before the <filter> parameter is processed. For example, if the <with-system> parameter is equal to

'report-all', then the <filter> parameter is conceptually applied to all data nodes and all system configuration data.

6.3.2. <edit-config> Operation

The <edit-config> operation has several editing modes. The 'create', and 'delete' editing operations are affected by the system configuration data handling basic mode. The other enumeration values for the NETCONF operation attribute are not affected.

If the operation attribute contains the value 'create', and the data node already exists in the target configuration datastore, then the server MUST return an <rpc-error> response with a 'invalid-value' error-tag.

If the client sets a data node that is explicitly set by the server, the server MUST accept the request if it is valid. The server MUST keep or discard the new value based on its system configuration data handling basic mode.

7. YANG Module for the <with-system> Parameter

The following YANG module defines the addition of the with-system element to the <get>, <get-config>, and <copy-config> operations. The YANG language is defined in [RFC6020]. The above operations are defined in YANG in [RFC6241]. Every NETCONF server which supports the :with-system capability MUST implement this YANG module.

```
<CODE BEGINS> file="ietf-netconf-with-system@2019-12-31.yang"
module ietf-netconf-with-system {
   namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-with-system";
   prefix ncws;
   import ietf-netconf { prefix nc; }

   organization
    "IETF NETCONF (Network Configuration Protocol) Working Group";

   contact
    "WG Web:   <http://tools.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>
     WG Chair:
     Editor:
                                                ";
   description
    "This module defines an extension to the NETCONF protocol
     that allows the NETCONF client to control how system configuration
     data are handled by the server in particular NETCONF operations.
```

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

revision 2019-12-31 {
  description
    "Initial version.";
  reference
   "RFC XXXX: With-system capability for NETCONF";
}

typedef with-system-mode {
   description
     "Possible modes to report system configuration data.";
   reference
      "RFC XXXX; section 3.";
      // RFC Ed.: replace XXXX with actual
      // RFC number and remove this note

   type enumeration {
      enum report-all {
          description
            "All system configuration data is reported.";
          reference
            "RFC XXXX; section 3.1";
            // RFC Ed.: replace XXXX with actual
            // RFC number and remove this note

      }
      enum report-all-tagged {
          description
            "All system configuration data is reported.
             Any nodes considered to be system configuration
             data will contain a 'origin' XML attribute,
             set to 'system'.";
          reference
            "RFC XXXX; section 3.4";
            // RFC Ed.: replace XXXX with actual
```

```
                // RFC number and remove this note
          }
          enum trim {
              description
                "Values are not reported if they contain the system
                configuration data.";
              reference
                "RFC XXXX; section 3.2";
                // RFC Ed.: replace XXXX with actual
                // RFC number and remove this note

          }
          enum explicit {
              description
                "Report values that contain the definition of
                 explicitly set data.";
              reference
                "RFC XXXX; section 3.3";
                // RFC Ed.: replace XXXX with actual
                // RFC number and remove this note
          }
      }
    }

    grouping with-system-parameters {
      description
        "Contains the <with-system> parameter for control
         of system configuration data in NETCONF retrieval
         operations.";

      leaf with-system {
        description
          "The explicit system configuration data processing
           mode requested.";
        reference
          "RFC XXXX; section 4.6.1";
          // RFC Ed.: replace XXXX with actual
          // RFC number and remove this note

        type with-system-mode;
      }
    }

    // extending the get-config operation
    augment /nc:get-config/nc:input {
        description
          "Adds the <with-system> parameter to the
           input of the NETCONF <get-config> operation.";
```

```
        reference
          "RFC XXXX; section 4.6.1";
          // RFC Ed.: replace XXXX with actual
          // RFC number and remove this note

        uses with-system-parameters;
    }

    // extending the get operation
    augment /nc:get/nc:input {
        description
          "Adds the <with-system> parameter to
           the input of the NETCONF <get> operation.";
        reference
          "RFC XXXX; section 4.6.1";
          // RFC Ed.: replace XXXX with actual
          // RFC number and remove this note

        uses with-system-parameters;
    }
}
<CODE ENDS>
```

8. IANA Considerations

This document registers the following capability identifier URN in the 'Network Configuration Protocol Capability URNs registry':

urn:ietf:params:netconf:capability:with-system:1.0

This document registers two XML namespace URNs in the 'IETF XML registry', following the format defined in [RFC3688].

```
      URI: urn:ietf:params:xml:ns:netconf:system:1.0
      URI: urn:ietf:params:xml:ns:yang:ietf-netconf-with-system
```

Registrant Contact: The NETCONF WG of the IETF.

XML: N/A, the requested URIs are XML namespaces.

This document registers one module name in the 'YANG Module Names' registry, defined in [RFC6020] .

```
      name: ietf-netconf-with-system
      prefix: ncws
      namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-with-system
      RFC: XXXX // RFC Ed.: replace XXXX and remove this comment
```

9. Acknowledgements

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

10.2. Informative References

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

Appendix A. Usage Examples

A.1. Example YANG Module

The following YANG module defines an example interfaces table to demonstrate how the <with-system> parameter behaves for a specific data model.

```
        container interfaces {
          list interface {
            key name;
            leaf name {
              type string;
            }
            leaf description {
              type string;
            }
            leaf-list ip-address {
              type inet:ip-address;
            }
          }
        }
```

A.2. Example Data Set

The following data element shows the conceptual contents of the example server for the protocol operation examples in the next section. This includes all the configuration data nodes and system configuration leafs.

```
   <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <interfaces xmlns="http://example.com/ns/interfaces">
       <interface>
         <name>lo0</name>
         <ip-address>127.0.0.1</ip-address>
         <ip-address>::1</ip-address>
       </interface>
       <interface>
         <name>lo1</name>
         <description>loopback</description>
         <ip-address>127.0.0.1</ip-address>
         <ip-address>::2</ip-address>
       </interface>
       <interface>
         <name>lo2</name>
         <description>loopback</description>
         <ip-address>127.0.0.1</ip-address>
         <ip-address>::3</ip-address>
       </interface>
       <interface>
         <name>lo3</name>
         <ip-address>127.0.0.1</ip-address>
         <ip-address>::1</ip-address>
       </interface>
     </interfaces>
     </data>
```

In this example, the 'ip-address' field for each interface entry is set in the following manner:

```
+---------+----------+-------+
| name    |ip-address| set by|
+---------+----------+-------+
| lo0     |127.0.0.1 | server|
| lo0     |  ::1     | server|
| lo1     |127.0.0.1 | server|
| lo1     |  ::2     | client|
| lo2     |127.0.0.1 | server|
| lo2     |  ::3     | client|
| lo3     |127.0.0.1 | server|
| lo3     |  ::1     | server|
+---------+----------+-------+
```

A.3. Protocol Operation Examples

The following examples show some <get> operations using the 'with-system' element. The data model used for these examples is defined in Appendix A.1.

The client is retrieving all the data nodes within the 'interfaces' object, filtered with the <with-system> parameter.

A.3.1. <with-system> = 'report-all'

The behavior of the <with-system> parameter handling for the value 'report-all' is demonstrated in this example.

```
      <rpc message-id="101"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get>
          <filter type="subtree">
            <interfaces xmlns="http://example.com/ns/interfaces"/>
          </filter>
          <with-system
           xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-system">
            report-all
          </with-system>
        </get>
      </rpc>

      <rpc-reply message-id="101"
                 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <data>
       <interfaces xmlns="http://example.com/ns/interfaces">
         <interface>
           <name>lo0</name>
           <ip-address>127.0.0.1</ip-address>
           <ip-address>::1</ip-address>
         </interface>
         <interface>
           <name>lo1</name>
           <description>loopback</description>
           <ip-address>127.0.0.1</ip-address>
           <ip-address>::2</ip-address>
         </interface>
         <interface>
           <name>lo2</name>
           <description>loopback</description>
           <ip-address>127.0.0.1</ip-address>
           <ip-address>::3</ip-address>
         </interface>
         <interface>
           <name>lo3</name>
           <ip-address>127.0.0.1</ip-address>
           <ip-address>::1</ip-address>
         </interface>
       </interfaces>
     </data>
     </rpc-reply>
```

A.3.2. <with-system> = 'report-all-tagged'

The behavior of the <with-system> parameter handling for the value 'report-all-tagged' is demonstrated in this example. A 'tagged' data

node is an element that contains the 'origin' XML attribute, set to 'system'.

The actual data nodes tagged by the server depend on the system configuration data handling basic mode used by the server. Only the data nodes that are considered to be system configuration data will be tagged.

In this example, the server's basic mode is 'explicit', then only data nodes that are not explicitly set data are tagged. If the server's basic mode is 'report-all', then no data nodes are tagged.

```
<rpc message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <interfaces xmlns="http://example.com/ns/interfaces"/>
    </filter>
    <with-system
     xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-system">
      report-all-tagged
    </with-system>
  </get>
</rpc>
```

```
       <rpc-reply message-id="102"
                  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
                  xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">
      <data>
        <interfaces xmlns="http://example.com/ns/interfaces">
          <interface or:origin="or:system">
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
          <interface>
            <name>lo1</name>
            <description>loopback</description>
            <ip-address or:origin="or:system">127.0.0.1</ip-address>
            <ip-address>::2</ip-address>
          </interface>
          <interface>
            <name>lo2</name>
            <description>loopback</description>
            <ip-address or:origin="or:system">127.0.0.1</ip-address>
            <ip-address>::3</ip-address>
          </interface>
          <interface or:origin="or:system">
            <name>lo3</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
        </interfaces>
      </data>
      </rpc-reply>
```

A.3.3. <with-system> = 'explicit'

The behavior of the <with-system> parameter handling for the value 'explicit' is demonstrated in this example.

```
      <rpc message-id="103"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get>
          <filter type="subtree">
            <interfaces xmlns="http://example.com/ns/interfaces"/>
          </filter>
          <with-system
           xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-system">
            explicit
          </with-system>
        </get>
      </rpc>
```

```
       <rpc-reply message-id="101"
                  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <data>
        <interfaces xmlns="http://example.com/ns/interfaces">
          <interface>
            <name>lo0</name>
          </interface>
          <interface>
            <name>lo1</name>
            <description>loopback</description>
            <ip-address>::2</ip-address>
          </interface>
          <interface>
            <name>lo2</name>
            <description>loopback</description>
            <ip-address>::3</ip-address>
          </interface>
          <interface>
            <name>lo3</name>
          </interface>
        </interfaces>
       </data>
     </rpc-reply>
```

A.3.4. <with-system> = 'trim'

The behavior of the <with-system> parameter handling for the value 'trim' is demonstrated in this example.

```
      <rpc message-id="104"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get>
          <filter type="subtree">
            <interfaces xmlns="http://example.com/ns/interfaces"/>
          </filter>
          <with-system
           xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-system">
            trim
          </with-system>
        </get>
      </rpc>

      <rpc-reply message-id="101"
                 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <data>
       <interfaces xmlns="http://example.com/ns/interfaces">
         <interface>
           <name>lo0</name>
           <ip-address>127.0.0.1</ip-address>
           <ip-address>::1</ip-address>
         </interface>
         <interface>
           <name>lo1</name>
           <description>loopback</description>
           <ip-address>127.0.0.1</ip-address>
         </interface>
         <interface>
           <name>lo2</name>
           <description>loopback</description>
           <ip-address>127.0.0.1</ip-address>
         </interface>
         <interface>
           <name>lo3</name>
           <ip-address>127.0.0.1</ip-address>
           <ip-address>::1</ip-address>
         </interface>
       </interfaces>
     </data>
     </rpc-reply>
```

Authors' Addresses

Feng Chong
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: frank.fengchong@huawei.com

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: maqiufang1@huawei.com