NETCONF
Internet-Draft
Intended status: Standards Track
Expires: December 24, 2021

C. Feng
Q. Ma
Huawei
C. Xie
China Telecom

June 22, 2021

System Configuration Data Handling Behavior
draft-ma-netconf-with-system-02

Abstract

This document defines an optional "system" datastore to allow clients populate the system configuration into running datastore in the device. It also defines a capability-based extension to the NETCONF protocol that helps the NETCONF client identify how system configuration are processed by the server.

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 https://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 December 24, 2021.

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 system generated data item (e.g., the auto-created interfaces "eth1"), which is only present in the <operational> datastore [RFC8342]. In order for it being referenced, the duplicated system configured data item needs to be retrieved from <operational> and overridden by the client.

In some other cases, a system generated configured data item is in the when/must statement, the similar operation should also be performed to make sure a successful validation, which is cumbersome.

Furthmore, when the system generated data item gets updated, there is no way to synchronize the update into <running> and the client can't detect the update automatically.

This document defines a "system" datastore to hold all the configurations provided by the system itself. It also defines a capability-based extension to the NETCONF protocol that allows the configuration synchronization between <system> and <running> both automatically and explicitly.

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 provided by the system itself [RFC8342].

System configuration datastore: A configuration datastore holding the complete configuration provided by the system iteself. This datastore is referred to as "<system>".

physical resource independent 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 without must/when statement constraint (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.

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 system 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. <system> does not persist across reboots. It will be automatically loaded when the device is powered on or the physical resource is present.

2.1. Life Cycle of the system configuration management

When the device is powered on, physical resource independent system configuration will be created in <system> automatically by the system if there is no when/must statement constraint associated with system configuration data or provided only when a special functionality is enabled.

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>;

2.2. "Factory-Reset" RPC Impact on System Configuration Datastore

[RFC8808]defines a "factory-reset" RPC to allow clients to reset a server back to its factory-default condition. Upon receiving the RPC, all supported conventional read-write configuration datastore(i.e.,<running>, <startup> and <candidate>) are reset to the contents of <factory-default>. <system> should also immediately reset to its factory default state. That's to say, any system configurations generated due to system upgrading or client-enabled functionality should be discarded. System configuration which is generated at the first time when it boots after being shipped from factory should be retained.

3. 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 advertise a particular style of system configuration data handling, and the client can adjust behavior accordingly.

This document specifies two standard system configuration handling basic modes that a server implementor may choose from:

- auto-populate
- no-populate

A server that uses the 'auto-populate' basic mode MUST automatically

- Update <running> with the system configuration change, after the "system" configuration has been altered as a consequence of a plug and play operation or device powering on operation. However the configurations in <running> can not be removed automatically when configuration data nodes in <system> is deleted since those configurations in <running> are likely to have already been modified or referenced.
- 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 'no-populate' 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.

3.1. 'auto-populate' Initialization During Reboot

The contents of <system> don't have to be persist across reboots, even in the presence of non-volatile storage.

For the NETCONF server which implements the <factory-default> datastore [RFC8808], it may load <factory-default> at the first time when it boots after being shipped from factory or reset to its factory default condition. Then it's just like each normal boot time, the device generates system configurations and saves into <system>. Then the device loads the saved startup configuration(if <startup> exists) into <running>. Lastly, the device loads <system> into <running>. If there exists any conflict, the configuration in the <running> should succeed.

3.2. 'auto-populate' <edit-config> Behavior towards <running>

For a data node that is loaded from <system> automatically, the server MUST consider it to exist.

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

3.3. 'no-populate' <edit-config> Behavior towards <running>

The server MUST NOT consider any system configuration data node to exist in <running> configuration datastore, except those explicitly set by the client.

- A valid 'create' operation attribute for a data node that is set by the server MUST succeed since the system configuration data is not present in the <running> configuration datastore.

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

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

4. Retrieval of System Configuration Data

TBD

5. With System Capability

5.1. Overview

The :with-system capability indicates which system-data-handling basic mode is supported by the server. These basic 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 6 for the system configuration data handling mode indicated by the 'basic-mode' parameter.

5.2. 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 3. The allowed values of this parameter are 'auto-populate', and 'no-populate', as defined in Section 3.

urn:ietf:params:netconf:capability:with-system:1.0?basic-mode=no-populate

5.3. Modifications to Existing Operations

5.3.1. <get-data> Operations

As defined in Section 6, retrieval of system configuration in <system> can be used through <get-data> operation with the "datastore" parameter set to "system".

5.3.2. <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.

6. YANG Module

This YANG module uses the "datastore" identity [RFC8342]. Every NETCONF server which supports :with-system capability MUST implement this YANG module.

```
<CODE BEGINS> file="ietf-netconf-with-system@2021-05-14.yang"
module ietf-netconf-with-system {
   yang-version 1.1;
   namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-with-system";
   prefix ncws;
```

```
    import ietf-datastores {
    prefix ds;
    reference
      "RFC 8342: Network Management Datastore Architecture(NMDA)";
    }

    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.

      Copyright (c) 2010 IETF Trust and the persons identified as
      the document authors.  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

    revision 2021-05-14 {
      description
        "Initial version.";
      reference
       "RFC XXXX: With-system capability for NETCONF";
    }

    feature system-datastore {
      description
        "Indicates that the system configuration is available as a datastore.";
    }

    identity system {
      if-feature "system-datastore";
```

```
      base ds:datastore;
      description
        "This read-only datastore contains the system configuration for the
         device that will be loaded into <running> automatically in the
         "auto-populate" basic mode.";
    }
 }
 <CODE ENDS>
```

7. 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
```

8. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

9. Acknowledgements

Thanks to Robert Wilton, Kent Watsen, Balazs Lengyel, Timothy Carey for reviewing, and providing important input to, this document.

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>.

[RFC8808] Wu, Q., Lengyel, B., and Y. Niu, "A YANG Data Model for Factory Default Settings", RFC 8808, DOI 10.17487/RFC8808, August 2020, <https://www.rfc-editor.org/info/rfc8808>.

Appendix A. Changes between Revisions

v01 - v02

- Remove System configuration data retrieval behavior in the mainbody and examples in the appendix.
- Remove <get> operation and <get-config> operation extension from the YANG data model.
- Change basic mode values into auto-populate, no-populate.
- Consider <factory-default> to work together with <system>.

Appendix B. Open Issues tracking

- Do we need to define RPC to allow the server loads <system> configuration data into <running>?
- Can we introduce better terminology?
- Should we define a standard operation of system configuration retrieval?

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

Chongfeng Xie
China Telecom
Beijing
China

Email: xiechf@chinatelecom.cn