Network Working Group
Internet Draft
Intended status: Standard Track
Expires: January 3, 2015

G. Zheng
X. Ji
Huawei Technologies

July 2, 2014

Integrating Operations in YANG Models
draft-zheng-netmod-integrate-operations-00.txt

Abstract

This document introduces an extension to YANG. The extension allows operation methods to be directly integrated in YANG models.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

YANG is a data modeling language used to model data manipulated by the NETCONF Protocol. YANG provides the capability to describe configuration/state data in a tree structure; and it has been proven that YANG is an efficient language to model data for network configuration.

However, along with management requirements of modern NMS becoming more and more sophisticate, the capability of only describing the data structure in the model seems to be insufficient in some scenarios. In a nutshell, these scenarios require some operations to be integrated in the model as well.

This document firstly describes the scenarios and then introduces a YANG model extension which allows operation method to be directly integrated in YANG models.

2. Requirements Language and Terminology

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 [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "should" or "Should"), they have their usual English meanings, and are not to be interpreted as [RFC2119] key words.

Terminology:

3. Scenarios and Requirements of Operation Extension in Data Models

In some datastore operation scenarios, the operations triggered from the NETCONF client might be a set of combined operations or a complex single operation within the NETCONF agent, which may change the datastores. And sometimes the datastore change might not be from client but from the agent itself. These operations need to manipulate the data in running datastore or candidate datastore, and thus they need the <edit-config> capability a well as a unified authorization system to control the influence to the data stores caused by them.

3.1. Scenarios of Complex Operations

The complex operations on data model such as batch creating or converting dynamic data to configuration data, are targeting the running datastore or candidate datastore. Thus, the target datastore, error-option, confirmed-commit, validation and other attributes of <edit-config> operation need to be specified. The following are some examples.

o Scenario 1: converting the device’s dynamic ARP to static ARP

The ARP data model contains static configuration attributes and dynamic non-configuration attributes; both of them are needed by ARP service. The difference between these two groups of attributes is that the static ARP is configured by the client while the dynamic ARP is learn from network peers. To accelerate the speed of dynamic ARP learning after rebooting device, or to simplify the configuration of the static ARP, the dynamic ARP entries need to be converted to static ARP entries when the network is stable.

Current solution 1:

The client gets the dynamic ARP data then uses these data as static ARP configuration and add them to the NETCONF agent by using <edit-config> operation.

Problems of solution 1:

- There are too many data communications between client and server, thus the performance is low.
- The configured ARP by dynamic ARP from client is duplicated with the dynamic ARP in the device, thus the device has to process the duplication.

Current solution 2:

According to [RFC6020]/[RFC6241] RPC definition, define a new non-standard RPC to convert dynamic ARP to static ARP directly.

Problems of solution 2:

The new RPC cannot use the existing attributes of <edit-config> operation, such as target datastore, error-option, confirmed commit and validate. The NETCONF agent has to implement these <edit-config> operation attributes itself, which makes the agent more complex.

o Scenario 2: batch operations on data model

In some situations, batch merge operations on the list of one module might be needed to reduce the communication data size.

Current solutions:

1) Use <get-config> with filter to get the data from the device, and then add all merge operations to <edit-config> and send it to the device.

2) According to [RFC6020]/[RFC6241] RPC definition, define a non-standard RPC to do the batch merge operation.

Problems of current solutions

1) Solution 1 impacts the operation efficiency between host and device significantly.

2) Solution 2 has the same issues with Use Case 1’s solution 2.

o Scenario 3: maintaining operations on data models

The data model of one module is usually hierarchical. Sometimes it needs maintaining operations on the sub nodes, such as resting the running services.

Current solution:

According to [RFC6020]/[RFC6241] RPC definition, define a non-standard RPC to do maintaining operations.

Problems of current solution:

1) When defining the RPC operation, all the key attributes of parents path should appear in the input parameter of the RPC, otherwise the instance of the sub node could not be located. The additional key parameters make the input parameter definition redundant.

2) The scope of sub nodes that the new defined RPC would impact is not clear, thus users have to refer other documents to do authorization rather than just referring the RPC definition itself.

3.2. Scenario of Operation Authorization

The user-defined non-standard RPC operations might impact the datastores, which need additional authorization.

Current solution:

When the user-defined RPC operation impact the datastores (either directly manipulate the datastores or impact the datastores as a side effect of the protocol operation), then the server MUST intercept the access operation and make sure the user is authorized to perform the requested access operation on the targeted data as defined in [RFC6536] Section 3.4.5.

Problems of Current Solution:

1) Because the scope of datastores that the RPC operation would impact is not defined exactly, there is a risk that invocation of non-standard protocol operations might have undocumented side effects ([RFC6536] Section 3.7.2).

2) An administrator needs to set access control rules to make the configuration datastore protected from user-defined non-standard protocol operation’s side effects. But, the rules definition has to depend on the relevant documents not directly on model definition, thus it is not the same as the data node access control rule definition, which makes the authorization management more complex.

3.3. The Requirements for Operation Extension in Data Models

The user-defined non-standard operation should overload the standard operations of <edit-config>, <get>, <get-config> and so on, so that the new operation could inherit the capabilities of the base standard operations.

The user-defined non-standard operations should be able to clearly define the scope of the datastores that the operations are allowed to impact in terms of build-in RPC semantics rather than outside documents.

4. Solution for Operation Extension in Data Models

4.1. General Idea

1) The operations to data nodes should be defined on the data model hierarchy tree as a part of the data model attributes; the data

node’s sub-statement of the operations should also be defined as well..

2) The operations that will effect or side effect datastores should be specified which base standard RPC operations are inherited.

3) When the data model operations are delivered by NETONCF, they MUST be in the format of the base protocol RPC operations; and the operations definition should be presented as a "Method" attribute of data node; the base protocol RPC operation capabilities should be inherited.

4) The operation access control rules should depend on the impact to data models by the operations defined in the model.

4.2. Usage Example

This example is based on the ARP conversion scenario described in Section 3.1.

Operation is defined as a "method" substatement in the YANG model:

```
module arp {

  namespace "http://example.com/network";

  prefix "arp";

  container arp-records {

     list arpList {

        leaf ipAddr {

           type string

        }

        leaf macAddr {

           type string

        }

        leaf styleType {

           type string;
        }

        method convert-arp  {

           base edit-config

           description "A method to convert dynamic arp to static
           arp";

           input {

              leaf source-arptype {

                 type string;

              }

              leaf dest-arptype {

                 type string;

              }

           }

        }

  }

}
```

Corresponding NETCONF operation is as the following:

```
   <rpc message-id="101"

        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"

        xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">

     <edit-config>

       <target>

         <running/>

       </target>
       <config>

         <arp xmlns="http://example.com/network">

           <arp-records method="convert-arp">

               <source-arptype>dynamic</source-arptype >

               <dest-arptype>dynamic</dest-arptype >

           </arp-records >

         </arp>

       </config>

     </edit-config>

   </rpc>
```

5. Security Considerations

TBD

6. IANA Considerations

This document requires no IANA registration.

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.

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

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, July 2013.

7.2. Informative References

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

8. Acknowledgments

Many valuable comments were received from Gang Yan to improve the draft. Bing Liu also participated in forming this draft.

This document was prepared using 2-Word-v2.0.template.dot.

Authors’ Addresses

Guangying Zheng
Huawei Nanjing R&D Center
Huawei Technologies Co., Ltd
101 Software Avenue, Yuhua District, Nanjing, Jiangsu
P.R.China. 210012

Email: zhengguangying@huawei.com

Xiaofeng Ji
Q14-4-B Building
Huawei Technologies Co., Ltd
Zhong-Guan-Cun Environmental Protection Park, No.156 Beiqing Rd.
Hai-Dian District, Beijing
P.R. China. 100095

Email: jixiaofeng@huawei.com