NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 5, 2016

K. Watsen
Juniper Networks
A. Bierman
Yumaworks
M. Bjorklund
Tail-f Systems
J. Schoenwaelder
Jacobs University Bremen

September 2, 2015

Operational State Enhancements for YANG, NETCONF, and RESTCONF

draft-kwatsen-netmod-opstate-00

Abstract

This document presents enhancements to YANG, NETCONF, and RESTCONF to better support the definition of and access to operational state data.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Support for operational state has been defined in YANG [RFC6020], NETCONF [RFC6241], and RESTCONF [draft-ietf-netconf-restconf] since their beginnings. However, after some operational experience, the support defined by these standards has been found to be limiting [draft-openconfig-netmod-opstate] as follows:

- YANG
  - Inability to associate operational state with configured state

- NETCONF
  - Inability to retrieve operational state without also retrieving running configuration
  - Inability to inspect the configuration as it is operationally running
- RESTCONF
  - Inability to inspect the configuration as it is operationally running

Addressing these limitations is the focus of this document.

2. Terminology

The following terms are defined in [draft-openconfig-netmod-opstate], but are redefined here as follows:

- intended configuration - this data represents the configuration state that the network operator intends the configuration controlled by the NETCONF/RESTCONF server to be in. In the NETCONF protocol, the intended configuration is specified in the "running" datastore. In the RESTCONF protocol, the intended configuration is specified in its conceptual datastore.

- applied configuration - this data represents the configuration state that the NETCONF/RESTCONF server is actually in, i.e., that which is currently being run by particular software modules (e.g., the BGP daemon), or other systems within the server (e.g., a secondary control-plane, or line card). The data model for applied configuration is the same as the intended configuration's data model. That is, the applied configuration data model is also defined by the config true nodes in YANG modules supported by the NETCONF/RESTCONF server. The data within the applied configuration is the same as the data within the intended configuration except as follows:

  - When the intended configuration has not been communicated to an external software entity

  - When post-processing or flattening of the intended configuration occurs to present a simpler view to the external software entities

  The transition from intended config to applied config commences in NETCONF when <edit-config> or <commit> is called, for :writable-

running or :candidate respectively, and in RESTCONF immediately whenever a POST, PUT, DELETE, or PATCH operation is called. Neither NETCONF nor RESTCONF currently enable inspection of the applied configuration.

- derived state - this data represents information which is generated as part of the system's own interactions. For example, derived state may consist of the results of protocol interactions (the negotiated duplex state of an Ethernet link), statistics (such as message queue depth), or counters (such as packet input or output bytes). Derived stated is defined in YANG using config false nodes, retrievable in NETCONF using the <get> RPC, and retrievable in RESTCONF using the content=nonconfiguration query parameter.

The following terms are defined in this document:

- intended state - a synonym for "intended configuration".

- operational state - the combination of applied state and derived state.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Conceptual Model

The following diagram illustrates the conceptual model presented in this document:

```
                                        +
                          intended      |    operational
                           state        |       state
                                        |
                                        |
                        +----------+    |    +---------+
     config true        | intended |    |    | applied |
     YANG nodes         |  config  |    |    | config  |
                        +----------+    |    +---------+
                                        |
   +--------------------------------------------------------+
                                        |
                                        |    +---------+
     config false                       |    | derived |
     YANG nodes                         |    |  state  |
                                        |    +---------+
                                        |
                                        +
```

Not illustrated in the diagram above:

- The intended and applied configurations share the same YANG-defined data model, specified by the config true nodes in the YANG modules supported by the server.

- The transition of the intended config to the applied commences immediately, whenever the intended config is updated.

4. Enhancements to YANG

4.1. The related-state Statement

The "related-state" statement identifies a path to where additional operational state associated for a config true node can be found. This operational state being in addition to any descendant config false nodes, which are implicitly associated to the parent config true node.

The "related-state" statement takes as an argument a string that is used to specify the path to a config false node holding the associated operational state. The format of the argument is the same as for the leafref's "path" statement, Section 9.9.2 in [RFC6020].

4.1.1. YANG Module

```
module ietf-yang-related-state {
  namespace urn:example:ietf-yang-related-state;
  prefix yrs;

  extension related-state {
    argument path;
    description
      "The related-state statement is used to identify a node that
       contains additional operational state associated for a config
       true node.

       The format of the argument is the same as for a leafref's "path"
       statement.

       The related-state statement can be specified in the following
       YANG statements:

         o  leaf
         o  leaf-list
         o  container
         o  list

       The related-state statement allows the following YANG substatements:

         o  description

       Multiple related-state statements can be given in a specific node.";
  }

}
```

4.1.2. Usage Example

The following example illustrates the related-state statement in use:

```
module ex-interfaces {
  namespace "http://example.com/interfaces";
  prefix xif;

  import ietf-yang-related-state {
    prefix yrs;
  }

  container interfaces {
    list interface {
      key name;
      yrs:related-state
        "/interfaces-state/interface[name=current()/name]";

      leaf name { type string }
      leaf mtu { type uint16; }
      ...
    }
  }
  container interfaces-state {
    config false;
    list interface {
      key name;
      leaf name { type string; }
      ...
    }
  }
}
```

5. Enhancements to NETCONF

5.1. The <get-state> Operation

One of the limitations identified in the Section 1 section was the inability for the NETCONF protocol to retrieve operational state without also retrieving running configuration. That is, the only defined NETCONF operation capable of returning operational state is the <get> operation ([RFC6241], Section 7.7), but it also returns the "running" configuration for the nodes selected by the passed filter. While it is possible to construct data-models whereby configuration and operational state are in completely isolated sub-trees, and thereby eliminate the retrieval of configuration when selecting an operational state node, requiring all models to be structured this way is not ideal.

5.1.1. YANG Module

```
   module ietf-netconf-get-state {
     namespace urn:example:ietf-netconf-get-state;
     prefix ncgs;

     import ietf-netconf {
       prefix nc;
     }

     rpc get-state {
       description
         "Retrieve device state information.";

       reference "RFC 6241, Section 7.7";

       input {
         anyxml filter {
           description
             "This parameter specifies the portion of the system
              configuration and state data to retrieve.";
           nc:get-filter-element-attributes;
         }
       }

       output {
         anyxml data {
           description
             "Copy of the running datastore subset and/or state
              data that matched the filter criteria (if any).
              An empty data container indicates that the request
              did not produce any results.";
         }
       }
     }
   }
```

5.2. The Applied Configuration Capability

5.2.1. Description

The applied configuration capability indicates that the device supports an applied configuration datastore, which is used to hold a read-only copy of configuration data as it is known to the operational components of the system (e.g., the data plane).

The applied configuration datastore contains applied configuration, as defined in Section 2.

5.2.2. Dependencies

None.

5.2.3. Capability Identifier

The :applied capability is identified by the following capability string:

```
:ietf:params:netconf:capability:applied:1.0
```

5.2.4. New Operations

None.

5.2.5. Modifications to Existing Operations

The :applied capability enables <applied/> to be passed as the <source> argument to the <get-config> and <copy-config> operations.

The :applied capability does not modify any other existing operations. In particular, the <applied/> value may not be used as the <target> argument to any operation.

Note, the :applied capability has no impact to the <get> operation because the <get> operation is defined as returning the "running" configuration, without any <source> parameter to specify otherwise.

The <applied/> parameter is formally defined in Section 5.2.6.

5.2.6. YANG Module

```
   module ietf-netconf-applied-config {
     namespace urn:example:ietf-netconf-applied-config;
     prefix ncac;

     import ietf-netconf {
       prefix nc;
     }

     augment /nc:get-config/nc:input/nc:source/nc:config-source {
       leaf applied {
          type empty;
       }
     }
     augment /nc:copy-config/nc:input/nc:source/nc:config-source {
       leaf applied {
          type empty;
       }
     }
   }
```

5.2.7. Example

To retrieve the "/interfaces" subtree from the applied configuration datastore:

```
   <rpc message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <get-config>
       <source>
         <applied xmlns="urn:example:ietf-netconf-applied-config"/>
       </source>
       <filter type="subtree">
         <interfaces xmlns="http://example.com/interfaces"/>
       </filter>
     </get-config>
   </rpc>
```

6. Enhancements to RESTCONF

6.1. The Applied Configuration Capability

6.1.1. Description

The applied configuration capability indicates that the device supports an applied configuration datastore, which is used to hold a read-only copy of configuration data as it is known to the operational components of the system (e.g., the data plane).

The applied configuration datastore contains applied configuration, as defined in section Section 2.

6.1.2. The applied capability

A RESTCONF server supports the applied configuration datastore when it presents the following URI in its "capability" leaf-list, as defined in [RFC6241], Section 9.3.

```
 urn:ietf:params:restconf:capability:applied:1.0
```

6.1.3. The "applied" Query Parameter

The "applied" parameter is only available when the RESTCONF server supports the "urn:ietf:params:restconf:capability:applied:1.0" capability.

The "applied" parameter is used to specify that the GET request should be directed to the applied configuration datastore.

The "applied" parameter does not have a value assignment.

This parameter is only allowed for GET methods on API, datastore, and data resources. A 400 Bad Request error is returned if it used for other methods or resource types.

7. Security Considerations

TBD

8. IANA Considerations

TBD

9. Acknowledgements

TBD

10. References

10.1. 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.

[draft-ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ieft-netconf-restconf-04 (work in progress), 2014, <https://tools.ietf.org/html/draft-ietf-netconf-restconf>.

10.2. Informative References

[draft-openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", 2015, <https://tools.ietf.org/html/draft-openconfig-netmod-opstate>.

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Andy Bierman
Yumaworks

EMail: andy@yumaworks.com

Martin Bjorklund
Tail-f Systems

EMail: mbj@tail-f.com

Juergen Schoenwaelder
Jacobs University Bremen

EMail: j.schoenwaelder@jacobs-university.de