Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 10, 2017

M. Bjorklund
Tail-f Systems
J. Schoenwaelder
Jacobs University
P. Shafer
K. Watsen
Juniper Networks
R. Wilton
Cisco Systems
May 9, 2017

Guidelines for YANG Module Authors (NMDA)
draft-dsdt-nmda-guidelines-01

Abstract

The "Network Management Datastore Architecture" (NMDA) adds the ability to inspect the current operational values for configuration, allowing clients to use identical paths for retrieving the configured values and the operational values. This change will simplify models and help modelers, but will create a period of transition as NMDA becomes a standard and is widely implemented. During that interim, the guidelines given in this document should help modelers find an optimal path forward.

Table of Contents

1. Introduction

This document provides advice and guidelines to help modelers plan for the emerging "Network Management Datastore Architecture" (NMDA) [I-D.ietf-netmod-revised-datastores]. This architecture provides an architectural framework for datastores as they are used by network management protocols such as NETCONF [RFC6241], RESTCONF [RFC8040] and the YANG [RFC7950] data modeling language. Datastores are a fundamental concept binding network management data models to network management protocols, enabling data models to be written in a network management protocol agnostic way.

1.1. Keywords

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].

1.2. Terminology

This document uses the terminology defined by the NMDA [I-D.ietf-netmod-revised-datastores].

1.3. Executive Summary

The Network Management Datastore Architecture (NMDA) addresses the so called "OpState problem" that has been the subject of much discussion in the IETF. NMDA is still in development and there will be a transition period before NMDA solutions are universally available.

These guidelines are aimed to enable the creation of models that can take advantage of the NMDA, while pragmatically allowing those models to be used with the existing network configuration protocol implementations.

It is the strong recommendation that models SHOULD move as quickly as possible to the NMDA. The specific approach to be taken for models being developed now and during the NMDA transition period should be based on both the expected usage and the maturity of the data model.

1. New models and models that are not concerned with the operational state of configuration information SHOULD immediately be structured to be NMDA-compatible.

2. Models that require immediate support for "in use" and "system created" information SHOULD be structured for NMDA. A non-NMDA version of these models SHOULD also be published, using either an existing model or a model created either by hand or with suitable tools that support current modeling strategies. Both the NMDA and the non-NMDA modules SHOULD be published in the same document, with NMDA modules in the document main body and the non-NMDA modules in a non-normative appendix. The use of the non-NMDA model will allow temporary bridging of the time period until NMDA implementations are available.

Additional details on these guidelines can be found below, notably in Section 2.

1.4. Background

NETCONF ([RFC6241]) was developed with a focus on configuration data, and has unfortunate gaps in its treatment of operational data. The <get-config> operation can return configuration data (defined as nodes with "config true") stored in <running>. This data is typically the only data created by CLI users and NETCONF clients. The <get> operation is defined as returning all the data on the device, including the contents of <running>, as well as any operational state data. While the NETCONF design envisioned models merging "config false" nodes with the contents of running, there are two issues involved.

First, the desire of clients to see the true operational ("in use") value of configuration data resulted in the need for data models to have two distinct leafs, one to show the configured value and the other to show the operational value. An example would be the speed of an interface, where the configured value may not be the value that is currently used.

Second, devices often have "system created" resources that exist as operational data even when there is no corresponding configuration data. An example would be built-in networking interfaces that always appear in operational data.

A similar situation to the second issue discussed above exists while the device is processing configuration data changes. When configuration data is deleted, the operational data will continue to exist during the time period in which the device is releasing resources associated with the data. An example would be deleting a BGP peer, where the peer continues to exist in operational data until the connection is closed and any other resources are released.

To address these issues without requiring a protocol modification, two distinct strategies have been adopted in YANG model design:

The first strategy makes two distinct top-level containers, one for configuration and one for state. These are sometimes referred to as "/foo" and "/foo-state". An example would be the interface model defined in [RFC7223]. These models require two completely distinct set of nodes, with repetition of both the interior containers, lists, and key nodes, but also repetition of many other nodes to allow visibility of the operational values of configured nodes. This leads to over-use of YANG groupings in ways that affect the readability of the models, as well as creating opportunities to incorrectly mirror the model's hierarchies. Also this "stitching together" of data from the two trees is merely a convention, not a formal relationship.

The second strategy uses two sibling containers, named "config" and "state", placed deeper within the model node hierarchy. The "config" container holds the configured values, while the "state" container holds the operational values. The duplication of interior nodes in the hierarchies is removed, but the duplication of leafs representing configuration remains. Groupings can be used to avoid the repetition of the leafs in the YANG file, but at the expense of readability. In addition, this strategy does not handle the existence of operational data for which there is no configuration data, such as the system-created data and deleted peers scenarios discussed above.

1.5. Network Management Datastores Architecture

The Network Management Datastores Architecture (NMDA) addresses the problems mentioned above by creating an architectural framework which includes a distinct datastore for operational data, called <operational>. This datastore is defined as containing both config true and config false nodes, with the formal understanding that the "in use" values are returned for the config true nodes. This allows modelers to use a single hierarchy for all configuration and operational data, which both improves readability and reduces the possibility of modeling inconsistencies that might impact programmatic access.

In addition, another datastore named <intended> is defined to provide a complete view of the configuration data, even in the presence of device-specific features that expand or remove configuration data. While such mechanisms are currently non-standard, the NMDA recognizes they exist and need to be handled appropriately. In the future, such mechanisms may become standardized.

The NMDA allows the deprecation of NETCONF's <get> operation, removing the source of these issues. The new operations <get-data> and <edit-data> will support a parameter indicating the target datastore. Similar changes are planned for RESTCONF ([RFC8040]).

2. Guidelines for YANG Modelers

The following guidelines are meant to help modelers develop YANG models that will maximize the utility of the model with both current implementations and NMDA-capable implementations. Any questions regarding these guidelines can be sent to yang-doctors@ietf.org.

The direction taken should be based on both the maturity of the data model, along with the number of concrete implementations of the model. The intent is not to destabilize the IETF modeling community, but to create models that can take advantage of the NMDA, while pragmatically allowing those models to be used with the existing network configuration protocol implementations.

It is the strong recommendation that models SHOULD move as quickly as possible to the NMDA. This is key to the future of these models. The NETMOD WG will rework existing models to this architecture. Given the permanence and gravity of work published by the IETF, creating future-proof data models is vital.

The two current strategies ("/foo-state" and "config"/"state" containers) mix data retrieval details into the data model, complicating the models and impairing their readability. Rather than

maintain these details inside the data model, models can be post-processed to add this derivative information, either manually or using tools.

Tools can automatically produce the required derived modules. The suggested approach is to produce a "state" version of the module with a distinct namespace, rather than using the "/foo-state" top-level container. Since the contents are identical, constraints in the data model such as "must" statements should not need to change. Only the model name, namespace, and prefix should need to change. This simplifies the tooling needed to generate the derived model, as well as reducing changes needed in client applications when transitioning to the NMDA model.

These derived models use distinct module names and namespaces, allowing servers to announce their support for the base or derived models.

Consider the following trivial model:

```
    module example-thermostat {
        namespace "tag:ietf:example:thermostat";
        prefix "thermo";

        container thermostat {
            leaf high-temperature {
                description "High temperature threshold";
                type int;
            }
            leaf low-temperature {
                description "Low temperature threshold";
                type int;
            }
            leaf current-temperature {
                description "Current temperature reading";
                type int;
                config false;
            }
        }
    }
```

In the derived model, the contents mirror the NMDA data model, but are marked as "config false", and the module name and namespace values have a "-state" suffix:

```
module example-thermostat-state {
    namespace "tag:ietf:example:thermostat-state";
    prefix "thermo-state";

    container thermostat {
        config false;
        leaf high-temperature {
            description "High temperature threshold";
            type int;
        }
        leaf low-temperature {
            description "Low temperature threshold";
            type int;
        }
        leaf current-temperature {
            description "Current temperature reading";
            type int;
        }
    }
}
```

By adopting a tools-based solution for supporting models that are currently under development, models can be quickly restructured to be NMDA-compatible while giving continuity to their community of developers. When NMDA-capable implementations become available, the base data models can be used directly.

Modelers and reviewers can view the simple data model, published in the body of document. Tools can generate any required derived models, and those models can be published in a non-normative appendix to allow interoperability.

It is critical to consider the following guidelines, understanding that our goal is to make models that will see continued use in the long term, balancing short term needs against a desire for consistent, usable models in the future:

(a) New models and models that are not concerned with the operational state of configuration information SHOULD immediately be structured to be NMDA-compatible.

(b) Models that require immediate support for "in use" and "system created" information SHOULD be structured for NMDA. A non-NMDA version of these models SHOULD exist, either an existing model or a model created either by hand or with suitable tools that mirror the current modeling strategies. Both the NMDA and the non-NMDA modules SHOULD be published in the same document, with NMDA modules in the document main body and the non-NMDA modules in a non-normative

appendix. The use of the non-NMDA model will allow temporary bridging of the time period until NMDA implementations are available.

(c) For published models, the model should be republished with an NMDA-compatible structure, deprecating non-NMDA constructs. For example, the "ietf-interfaces" model in [RFC7223] will be restructured as an NMDA-compatible model. The "/interfaces-state" hierarchy will be marked "status deprecated". Models that mark their "/foo-state" hierarchy with "status deprecated" will allow NMDA-capable implementations to avoid the cost of duplicating the state nodes, while enabling non-NMDA-capable implementations to utilize them for access to the operational values.

(d) For models that augment models which have not been structured with the NMDA, the modeler will have to consider the structure of the base model and the guidelines listed above. Where possible, such models should move to new revisions of the base model that are NMDA-compatible. When that is not possible, augmenting "state" containers SHOULD be avoided, with the expectation that the base model will be re-released with the state containers marked as deprecated. It is RECOMMENDED to augment only the "/foo" hierarchy of the base model. Where this recommendation cannot be followed, then any new "state" elements SHOULD be included in their own module.

3. IANA Considerations

This document has no actions for IANA.

4. Security Considerations

This document has no security considerations.

5. Informative References

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-01 (work in progress), March 2017.

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

[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, <http://www.rfc-editor.org/info/rfc6241>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

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

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <http://www.rfc-editor.org/info/rfc8040>.

Authors' Addresses

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Juergen Schoenwaelder
Jacobs University

Email: j.schoenwaelder@jacobs-university.de

Phil Shafer
Juniper Networks

Email: phil@juniper.net

Kent Watsen
Juniper Networks

Email: kwatsen@juniper.net

Rob Wilton
Cisco Systems

Email: rwilton@cisco.com