Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2017

R. Wilton, Ed.
Cisco Systems

July 7, 2016

Refined YANG datastores
draft-wilton-netmod-refined-datastores-01

Abstract

The existing definition of YANG datastores supported by NETCONF only provides a loose definition of the running datastore, and no formal definition of any datastore that represents the operational state of a device. This document defines a refined datastore model with new concrete and abstract datastores to allow devices to provide a clean separation between an operator's intended configuration of a device and the actual running operational state of a device. It provides a similiar, but alternative, datastore framework to the one described in draft-schoenw-netmod-revised-datastores.

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 http://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 January 8, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

This document defines a similar, but alternative, architectural datastore framework to [I-D.schoenw-netmod-revised-datastores]. The aim of this document is to make it easier to compare the models and approaches being described in the two different drafts. The reader is advised to also read [I-D.schoenw-netmod-revised-datastores] which provides a good background on why a refined NETCONF/YANG datastore model is needed.

This document defines serveral new datastores, and also introduces the new concept of an abstract datastore. Despite multiple new datastores being defined, the expectation is that clients and devices would mainly interact with only the Persistent Configuration and Operational State Datastores. Candidate and Startup datastores would be used as presently defined, and the Running datastore would be supported (as much as possible) for backwards compatibility purposes.

Abstract datastores are a new flavor of datastore that are semantically equivalent to regular datastores, but without the expectation that clients would be able to explicitly interact with them as explicitly named datastores. Instead, clients would likely infer the contents of the abstract datastores through metadata annotations on the regular datastores. One of the main advantages for defining these abstract datastores is to allow for a precise definition of the meaning of the configuration and operational data on a device, and potentially allow management agents to make comparisons between the different datastores. E.g. to determine if any intended configuration has not actually been applied, or perhaps conversely which parts of the applied configuration do not match the intended configuration.

This document also gives an idea of how ephemeral state could potentially be represented to meet the I2RS requirements specified in [I-D.ietf-i2rs-ephemeral-state]. Ephemeral configuration is treated as a separate optional configuration datastore that constitutes part of the intended configuration of the device. Ephemeral operational state is represented as part of the Operational State Datastore described in Section 5.8. Further refinement of the proposed handling of ephemeral state is likely needed to ensure that all the I2RS ephemeral state requirements are met.

2. Objectives

The key aims of the datastore definitions given in this document are:

to keep the existing definition of the running-configuration unchanged,

to minimize the impact on existing NETCONF/RESTCONF implementations, and to provide a viable upgrade path for existing NETCONF/RESTCONF servers,

to minimize the number datastores (and amount of data) that need to be explicitly managed by external management agents,

to make explicit the meaning of the contents of each datastores and how they relate to each other.

3. Definitions

The following terms are defined in [I-D.ietf-netmod-opstate-reqs]:

Intended Configuration

Applied Configuration

Operational State

The following definition is taken from [RFC6241]:

running configuration datastore - A configuration datastore holding the complete configuration currently active on the device. The running configuration datastore always exists.

The following new terms are defined here:

opstate aware device - a device that implements the requirements specified in [I-D.ietf-netmod-opstate-reqs], in particular the device must expose the split between the device's 'intended configuration' vs 'applied configuration'.

opstate unaware device - a device that is not an 'opstate aware device'. In particular, it does not draw a clear distinction between 'intended configuration' vs 'applied configuration', and generally treats them as having exactly the same contents.

abstract datastore - a conceptual type of datastore that represents some abstract property (e.g. 'applied configuration') of the data nodes that it contains. Although devices could allow

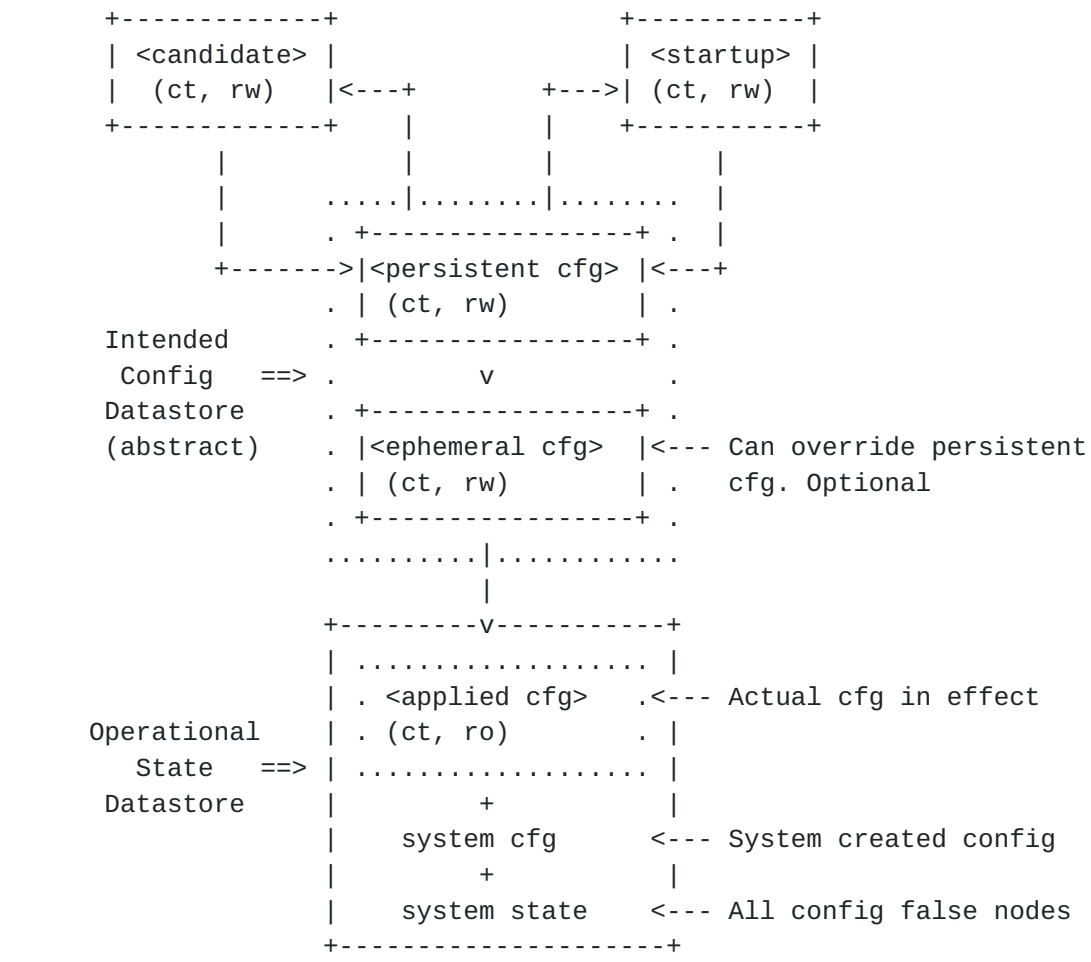
an abstract datastore to be externally referenced as a named datastore, this is not expected or required.

4. Overview of a refined model of datastores

This document defines a refined datstore model that can universally be used for both opstate aware devices and also existing NETCONF/ RESTCONF servers. The model is intended to cover both the opstate requirements [I-D.ietf-netmod-opstate-reqs] and the I2RS ephemeral configuration datastore requirements [I-D.ietf-i2rs-ephemeral-state].

All datastores described in this document use the same YANG schema, although the actual data nodes that are allowed in a particular datastore can depend on statements set in the schema. For example, the configuration datastores only contain datanodes for YANG schema nodes that are "config=true".

The following diagram illustrates how the datastores (except 'Running') relate to each other:



Key
Solid boxes (-----) indicate normal datastores:
(i.e Startup, Persistent Cfg, Ephemeral Cfg, Operational State)

Dotted boxes (.....) indicate abstract datastores:
(i.e. Intended Config and Applied Config)

ct = config true, rw = read/write, ro = read/only

Three new datastores are defined:

- Persistent Configuration - holds the current, client provided, configuration for the device that is saved to the Startup Datastore and is recovered after device reboot.

- Ephemeral Configuration - an optional datastore that holds client provided transient configuration that is discarded after device reboot.

- Operational State - a read-only datastore that holds all of the operational state of the device. Specifically it holds: the exact configuration that has been applied, along with any system controlled configuration, and all system state (including statistics and any ephemeral state).

Two new abstract datastores are defined:

- Intended Configuration - represents the combined desired configuration of the device

- Applied Configuration - represents the actual applied configuration of the device

Two datastores are unchanged from the NETCONF [RFC6241] definition:

- Candidate - represents candidate configuration

- Startup - represents startup configuration

For opstate aware devices, the Running Configuration Datastore is redefined as an abstract datastore that represents the combined persistent and applied configuration. It is regarded as a logical expansion of the definition in NETCONF [RFC6241].

5. Definition of all datastores

5.1. Candidate Datastore (optional)

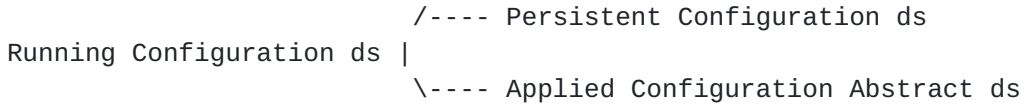
Holds candidate configuration. Unchanged from NETCONF [RFC6241].

5.2. Startup Datastore

Holds the saved configuration that is used by the device after reboot. Unchanged from NETCONF [RFC6241].

5.3. Running Configuration Datastore

To accommodate a clean separation between configuration and state, for an opstate aware device, the Running Configuration Datastore is logically split into two component parts, the Persistent Configuration Datastore and Applied Configuration Datastore, as illustrated by the following diagram:



The Applied Configuration Abstract Datastore is part of the Operational State Datastore.

5.4. Persistent Configuration Datastore

The Persistent Configuration Datastore holds the current configuration provided by a client that is expected to be persisted over device reboot. The datastore can be read and written by a client. Any edit operations on the datastore are validated as per YANG constraints validation before being processed further. The persistent configuration datastore interacts with both the Candidate and Startup datastores, and forms part of the Intended Configuration Abstract Datastore.

5.5. Ephemeral Configuration Datastore (Optional)

The Ephemeral Configuration Datastore may optionally be supported to hold any configuration that must not persist over device reboot. This writable datastore could be regarded as a pane of glass overlay on the persistent configuration datastore, allowing nodes in the ephemeral configuration to override, or depend on, nodes in the persistent configuration if required.

A mechanism is required to allow a client to choose which values take precendence if a leaf with different values exists in both the persistent configuration and ephemeral configuration datastore.

Multiple layers of ephemeral configuration within the ephemeral datastore could be supported.

5.6. Intended Configuration Abstract Datastore

The Intended Configuration Abstract Datastore represents the entire combined intended configuration for the device. It is implemented as the net combination of the Persistent Configuration Datastore combined with the optional Ephemeral Configuration Datastore.

For devices that do not support ephemeral configuration, the Intended Configuration Abstract Datastore is exactly the same as the Persistent Configuration Datastore.

5.7. Applied Configuration Abstract Datastore

The Applied Configuration Abstract Datastore is the subset of the config true datanodes in the Operational State Datastore that represents the applied configuration on the device.

If the intended configuration has been completely successfully applied, then the contents of the Applied Configuration Abstract Datastore exactly matches the Intended Configuration Abstract Datatstore, conforming to the behaviour specified in [I-D.ietf-netmod-opstate-reqs].

5.8. Operational State Datastore

The Operational State Datastore represents all of the operational state of the device, and is made up of the applied configuration, along with any system controlled configuration, and all of the system state. It includes statistics and ephemeral state (both applied configuration and operational state nodes).

This datastore contains all nodes defined in the YANG schema (i.e. both config true and config false nodes). The contents of this datastore can only be updated by the device, and as such, from a client perspective this datastore is read only.

The data held in the Operational State Datastore can be further classified into five categories:

5.8.1. Applied Configuration

The Operational State Datastore contains the applied configuration that represents the configuration that the device is actual using to operate the device.

If the intended configuration has been completely successfully applied, then the applied configuration data nodes exactly matches the logical contents of the Intended Configuration Abstract Datatstore.

5.8.2. System Controlled Configuration

In addition to the applied configuration, the Operational State Datastore also contains any System Controlled Configuration data nodes. These data nodes, using the same YANG config true schema nodes as for the applied configuration, represent all configuration that is created and controlled by the system independently of the applied configuration. E.g. this would include system created interfaces, which exist in the Operational State Datastore regardless of whether they have been explicitly configured by a client.

There is no YANG schema keyword required to identify nodes that may be system controlled. Hence, a device could choose for any config true node in the YANG schema to be system controlled, but a device would be expected to identify to a client the data nodes in the

Operational State Datastore that are system controlled through a mechanism such as YANG Metadata (e.g. as described in [I-D.wilton-netconf-opstate-metadata]).

5.8.3. System State

System State represents all of the data nodes in the Operational State Datastore that are represented by config false nodes in the YANG schema.

5.8.4. Statistics

Statistics are a subset of the data nodes in the Operational State Datastore. Statistics nodes are identified in the YANG schema by the "statistics true" statement, that must also be identified as "config false".

5.8.5. Ephemeral State

Ephemeral state is the subset of the schema in the Operational State Datastore that represents ephemeral state nodes, which are identified in the YANG schema by the ephemeral keyword, including both the applied ephemeral configuration (config true, ephemeral true), and ephemeral operational state (config false, ephemeral true).

6. Discussion points

6.1. A complete and accurate representation of a device's configuration

Sometimes a device cannot completely implement all of the nodes and values specified by a YANG schema. Ideally a well behaved device would indicate which parts of the schema it cannot completely support via YANG deviations. But deviations do not work in all scenarios - support for particular values of a configuration leaf may be dependent on the underlying hardware that is present in the device at the time. In this and other similar scenarios, to ensure that a client can properly manage a device, the applied configuration must be a complete and accurate representation of all of the configuration that a device is actually running. This must include any features that are implicitly enabled by default without any client configuration, or places where the applied configuration value differs from the intended configuration value (e.g. perhaps to protect the system from being overloaded).

6.2. Missing resources

Configuration that cannot be applied because the system resources are missing (or have been exhausted) would logically exist in the intended configuration datastore but not in the applied configuration datastore.

As defined in [I-D.ietf-netmod-opstate-reqs], by default, a NETCONF or RESTCONF configuration commit would be expected to fail if some of the configuration was for system resources that were not present. Extensions to NETCONF and RESTCONF could be provided to allow for more control over configuration operations that contain configuration for system resources that are missing.

6.3. System controlled resources

System controlled resources (i.e. those resources that would exist in the system regardless of configuration) always exist as nodes in the Operational State Datastore as part of the "system controlled configuration". If the resources have also been configured then they would also be present in the abstract intended datastore, and if the configuration successfully applied, the abstract applied configuration datastore as well.

Clients could find out which nodes in the operational state datastore are system controlled by using YANG Metadata, e.g. as described in [I-D.wilton-netconf-opstate-metadata].

6.4. Auto-configured or auto-negotiated values

The applied configuration in the Operational State Datastore only represents the configuration that is applied, and is bound by the constraint that if the configuration has been successfully applied then it must exactly match the intended configuration. Hence, this generally requires that separate config false leaves in the Operational State Datastore are required to represent the exact values programmed in the device.

In the specific case that the operational value meets the following three constraints then no separate config false leaf is required:

1. it is directly controlled by configuration,

2. it has exactly the same schema value space as the corresponding configuration leaf, and

3. if configured, the operational value of the system matches the applied configuration.

This optimization is allowed because the config false leaf value in the Operational State Datastore would always have exactly the same lifecycle existence and value as the corresponding config true leaf representing the applied configuration in the Operational State Datastore.

6.5. Operational State with Different Origins

The definition of the Operational State Datastore is designed to allow config false leaves to depend on either explicitly configured entities (in the applied configuration datastore) or system created configuration entries. This definition facilitates the merging of separate configuration and state parts of YANG models into the same container/lists if desired.

An example are IP addresses of an interface that can originate from configuration, from DHCP, or may be dynamically auto-configured. In this case, the operational state datastore would contain all IP addresses. The explicitly configured addresses would logically exist in the abstract applied configuration datastore. Addresses learned through DHCP or dynamically configured would logically exist as system controlled config true data nodes in the Operational State Datastore. Enhanced get/notification requests with YANG metadata annotations could be used to amend the reply/notification with metadata information to indicate which datastore the entries logically exist in.

6.6. Statistics

Both the Overview of 2002 IAB Network Management Workshop [RFC3535] and NETCONF/YANG Network Management Architecture [RFC6244] categorises the state of a devices separately into configuration, operational state, and statistics. NETCONF and YANG have historically only categorised the state of a device into configuration and non-configuration.

This document considers statistics to be part of the Operational State Datastore, which is consistent with how statistics information is returned in current NETCONF/RESTCONF requests, and allows all operational state to be efficiently and easily retrieved together in one request. It is envisaged that YANG schema data nodes could be labelled with a new "statistics true" statement to allow for easy filtering of statistics data for NETCONF/RESTCONF GET/GET-CONFIG requests and also YANG pub/sub. I.e. the extensions should allow for requests against the Operational State Datastore that exclude statistics, along with requests that only include statistics (along with any necessary containing structure and keys).

6.7. YANG Defaults Handling

The protocol handling of YANG defaults is described in NETCONF With-defaults [RFC3535] and RESTCONF [I-D.ietf-netconf-restconf]. These documents allow a device to report how YANG defaults are normally handled in requests for data resources, but with the expectation that the same semantics apply to all datastores. There is no way to express that the default handling semantics should depend on the datastore that a request pertains to.

When considering operational state, the most useful semantics for handling YANG defaults depend on the particular datastore and what system data it represents. To allow servers to provide optimal default handling, it is proposed that an extension to, or new version of, With-defaults is defined to support the proposed semantics below:

For configuration datastores that directly represent client provided configuration (i.e. the Persistent Configuration Datastore, Ephemeral Configuration Datastore and Intended Configuration Abstract Datastore), the most useful semantics are to return the exact data nodes set by the client (i.e. this is equivalent to the With-defaults "explicit" capability mode). Using this mode makes it easy for clients to check that a device has received and is acting on the exact desired configuration. Further, clients can always strip default values out of the configuration that they send to the device if they wish.

For the Operational State Datastore, which represents the exact operational state of the device, it is most helpful if the device returns the exact operational state of the device including any data nodes with a value that matches the YANG schema default value (i.e. this is equivalent to the With-defaults "report-all" capability mode). The benefits to the client of being able to rely on the values provided by the device outweigh the slight increase in data and processing overhead.

In addition, it is desirable if that the new with-defaults semantics apply to both explicit NETCONF/RESTCONF GET/GET-CONFIG requests and also YANG pub/sub. In all cases, for servers that support the YANG With-defaults extension, clients can overide the default handling behavior to whichever semantics they desire.

7. Implications of the Refined Datastore Model

7.1. Implications for YANG

The proposed revised datastore definitions have the following implications on YANG:

- A new "statistics true|false" statement is added to the YANG language to label schema data nodes that only contain statistics:
  - Only "config false" schema data nodes may be labelled "statistics true".
  - Schema data nodes that are labelled "statistics true" may not have any decendent child nodes that are either labelled "config true" or "statistics false".
- A new "ephemeral true|false" statement is added to the YANG language to label schema data nodes that contain ephemeral state:
  - Schema data nodes labelled "config true" or "config false" may also be labelled "ephemeral true".
  - Schema data nodes labelled "statistics true" or "statistics false" may also be labelled "ephemeral true".
  - Schema data nodes labelled "config true" and "ephemeral true" cannot appear in the Persistent Configuration Datastore.
  - Schema data nodes that are labelled "ephemeral true" cannot have any decendent YANG schema nodes that are labelled "ephemeral false".

7.2. Implications for NETCONF

The proposed revised datastore definitions have the following implications on NETCONF:

- Support for the new configuration datastores is added to NETCONF:
  - <get> and <get-config> requests are supported on the new datastores, but both return the same data.
  - Only the Persistent Configuration and Ephemeral Configuration (along with Candidate/Startup) datastores are writable by clients.
  - It is an implementation choice whether devices support Intended Config and Applied Config as named, client accessable, datastores.

    * A new NETCONF capability would indicate which new configuration datastores are supported by the device.

- Support for the new Operational State Datastore is added to NETCONF:

    * <get> requests return all the data from the datastore.

    * <get-config> requests only return config true data nodes (applied config and system controlled config) from the datastore.

    * <edit-config> requests are not supported.

    * A new NETCONF capability would indicate that the device supports the Operational State Datastore.

7.2.1. Implications for Opstate Aware Devices

The following implications are specific to opstate aware devices supporting NETCONF:

- Configuration requests on the Running Configuration Datastore are handled as follows:

    * <edit-config> and <get-config> requests are mapped to the Persistent Configuration Datastore.

    * <get> requests are mapped to the Operational State Datastore.

7.2.2. Implications for Opstate Unaware Devices

One of the key aims of the refined datastore model described in this draft is to minimize the impact on existing (or opstate unaware) NETCONF/RESTCONF clients and devices. The assumption of this model is that for an opstate unaware device, the Persistent Configuration, Intended Configuration and Applied Configuration Datastores all hold exactly the same values, and are collectively labelled as the Running Configuration Abstract Datastore).

An opstate unaware device does not have to make any changes, but a device could add support for the following to maximize interoperability:

- All config requests on the Persistent Configuration, Intended Configuration, and Applied Configuration datastores can be mapped to the Running Configuration Datastore.

- If new YANG modules are supported that allow configuration and state nodes to be combined via the creation of system controlled entries, then <get> requests on the Running Configuration Datastore would also include any system controlled configuration entries along with decendent children nodes.

- The Operational State Datastore could be supported (for <get> and <get-config> requests only) and mapped to the Running Configuration Datatstore + config false nodes.

7.3. Implications for RESTCONF

The proposed revised datastore definitions have the following implications on RESTCONF:

- Support for explicit datastores is added to RESTCONF:
  - A new URL would be provided to allow for datastore specific access (e.g. "/restconf/ds/<datastore>/"
  - GET requests are supported on the new datastores, the content parameter could also be supported, but is pretty meaningless.
  - PUT, POST and PATCH are supported on the Persistent Config Datastore and Ephemeral Config Datastores, which are writable by clients.
  - It is an implementation choice whether devices support Intended Config and Applied Config as named, client accessable, datastores.
  - A new RESTCONF capability would indicate which new configuration datastores are supported by the device.
- Support for the new Operational State Datastore is added to RESTCONF:
  - GET requests return all the data from the datastore, and the content parameter used to choose whether config true, config false, or all nodes are returned.
  - PUT, POST and PATCH requests are not supported.
  - A new RESTCONF capability would indicate that the device supports the Operational State Datastore.
- Requests on the default RESTCONF datastore URL ("/restconf/data"):

* PUT, POST and PATCH requests are handled as if they were made on the Persistent Configuration Datastore.

* GET requests would be handled as if they were made on the Operational State Datastore.

7.3.1. Implications for Opstate Unaware Devices

An opstate unaware device does not have to make any changes, but a device could add support for the following to maximize interoperability:

- Support could be added for the named Persistent Configuration, Intended Configuration, and Applied Configuration datastores which would be handled as if the request was made directly on "/restconf/data".

- If new YANG modules are supported that allow configuration and state nodes to be combined via the creation of system controlled entries, then GET requests on "/restconf/data" would also include any system controlled configuration entries along with decendant children nodes.

- The Operational State Datastore could be supported for GET requests, and mapped to "/restconf/data".

8. Changes from previous version

Changes from previous versions:

- Changes from version -00:

  * The entire draft has been quite heavily restructured with an effort to make it easier to read

  * A definition of "abstract datastores" is given, with usage restricted to intended configuration, applied configuration, and running configuration

  * "System Controlled Configuration" and "System Controlled State" are no longer modelled as separate abstract datastores

  * The main diagram has been updated to make the relationship between the datastores clearer and more simple

  * Added support for schema labelling of "statistics" data nodes as part of the Operational State Datastore

* Added support for schema labelling of "ephemeral state" data nodes as part of the Operational State Datastore

9. Acknowledgements

This document is not solely the authors own work, but instead represents one view from the discussions to find a consensual solution to the operational state problem. It takes ideas from many people and uses parts of solutions described in the various internet drafts listed below. In particular, this document is an alternative to the revised datastore model described in draft-schoenw-netmod-revised-datastores [I-D.schoenw-netmod-revised-datastores], and reuses some of the structure and text from that document.

Work from the following internet drafts have helped form the basis of the datastore solution described in this draft:

- draft-bjorklund-netmod-operational [I-D.bjorklund-netmod-operational]
- draft-ietf-netmod-opstate-reqs [I-D.ietf-netmod-opstate-reqs]
- draft-kwatsen-netmod-opstate [I-D.kwatsen-netmod-opstate]
- draft-openconfig-netmod-opstate [I-D.openconfig-netmod-opstate]
- draft-schoenw-netmod-revised-datastores [I-D.schoenw-netmod-revised-datastores]
- draft-wilton-netmod-opstate-yang [I-D.wilton-netmod-opstate-yang]
- draft-ietf-i2rs-ephemeral-state [I-D.ietf-i2rs-ephemeral-state]

The following people were authors to these Internet-Drafts or otherwise actively involved in the discussions that led to this document:

- Lou Berger, Labn Consulting, <lberger@labn.net>
- Andy Biermann, YumaWorks, <andy@yumaworks.com>
- Martin Bjorklund, Tail-f Systems, <mbj@tail-f.com>
- Susan Hares, Huawei, <shares@ndzh.com>
- Jeff Haas, Juniper Networks, <jhaas@juniper.net>
- Marcus Hines, Google, <hines@google.com>

- Christian Hopps, Deutsche Telekom, <chopps@chopps.org>
- Acee Lindem, Cisco Systems, <acee@cisco.com>
- Ladislav Lhotka, CZ.NIC, <lhotka@nic.cz>
- Thomas Nadeau, Brocade Networks, <tnadeau@lucidvision.com>
- Juergen Schoenwaelder, Jacobs University Bremen <j.schoenwaelder@jacobs-university.de>
- Anees Shaikh, Google, <aashaikh@google.com>
- Rob Shakir, Jive Communications, <rjs@rob.sh>
- Kent Watsen, Juniper Networks, <kwatsen@juniper.net>

The author would also like the thank the following people who have kindly provided feedback on this draft: Matt Hall, Einar Nilsen-Nygaard.

10. IANA Considerations

None

11. Security Considerations

This document discusses a conceptual model of datastores for network management using NETCONF/RESTCONF and YANG. It has no security impact on the Internet.

12. References

12.1. Normative References

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-14 (work in progress), June 2016.

[I-D.ietf-netmod-opstate-reqs]
Watsen, K. and T. Nadeau, "Terminology and Requirements for Enhanced Handling of Operational State", draft-ietf-netmod-opstate-reqs-04 (work in progress), January 2016.

[RFC3535] Schoenwaelder, J., "Overview of the 2002 IAB Network Management Workshop", RFC 3535, DOI 10.17487/RFC3535, May 2003, <http://www.rfc-editor.org/info/rfc3535>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[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>.

[RFC6243] Bierman, A. and B. Lengyel, "With-defaults Capability for NETCONF", RFC 6243, DOI 10.17487/RFC6243, June 2011, <http://www.rfc-editor.org/info/rfc6243>.

[RFC6244] Shafer, P., "An Architecture for Network Management Using NETCONF and YANG", RFC 6244, DOI 10.17487/RFC6244, June 2011, <http://www.rfc-editor.org/info/rfc6244>.

[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>.

12.2. Informative References

[I-D.bjorklund-netmod-operational]
Bjorklund, M. and L. Lhotka, "Operational Data in NETCONF and YANG", draft-bjorklund-netmod-operational-00 (work in progress), October 2012.

[I-D.ietf-i2rs-ephemeral-state]
Haas, J. and S. Hares, "I2RS Ephemeral State Requirements", draft-ietf-i2rs-ephemeral-state-10 (work in progress), June 2016.

[I-D.kwatsen-netmod-opstate]
Watsen, K., Bierman, A., Bjorklund, M., and J. Schoenwaelder, "Operational State Enhancements for YANG, NETCONF, and RESTCONF", draft-kwatsen-netmod-opstate-02 (work in progress), February 2016.

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

[I-D.schoenw-netmod-revised-datastores]
Schoenwaelder, J., "A Revised Conceptual Model for YANG Datastores", draft-schoenw-netmod-revised-datastores-01 (work in progress), June 2016.

[I-D.wilton-netconf-opstate-metadata]
Wilton, R., "Refined YANG datastores with Meta-data", draft-wilton-netconf-opstate-metadata-00 (work in progress), July 2016.

[I-D.wilton-netmod-opstate-yang]
Wilton, R., ""With-config-state" Capability for NETCONF/RESTCONF", draft-wilton-netmod-opstate-yang-02 (work in progress), December 2015.

Author's Address

Robert Wilton (editor)
Cisco Systems

Email: rwilton@cisco.com