NMDA AND CONFD

tail-f a Cisco company

Table of Contents

1. Introduction

The datastore is the core concept which ties together YANG data models and protocols such as NETCONF and RESTCONF. Network Management Datastore Architecture (NMDA) is an updated datastore architecture defined by the IETF in RFC 8342. In addition to the previously existing datastores of <startup>, <running>, and <candidate>, NMDA introduces two new datastores, <intended> and <operational>. In addition to the new datastores, NETCONF has defined new RPCs and RESTCONF has new extensions to support the updated datastore architecture defined by NMDA.

This application note will explain some of the reasons for the new datastore architecture as well look at NMDA is and how it has been implemented as of ConfD 7.3.

2. The Original Datastore Architecture

The first NETCONF specification, RFC 4741, defined the three datastores, <startup>, <running>, and <candidate>. Of the three datastores, <running>, which represents the running configuration, was the only mandatory datastore.

The <startup> datastore could be used by the NETCONF server if the device has a startup configuration that is applied when the device is initially started or restarted.

The <candidate> datastore can be used to hold configuration data which can be manipulated without affecting the running configuration. Once the configuration data in the <candidate> datastore is finalized, it can be transactionally committed to the running configuration.

Note that each of these datastores holds configuration data. Although YANG introduced the ability to identify which parts of the model are configuration (“config true”) and operational (“config false”), there was no datastore defined (at least conceptually) to hold “config false” data. Keep in mind that we are discussing the NETCONF standard here and not ConfD’s implementation of CDB which includes an operational datastore. NETCONF had the <get-config> RPC to retrieve only configuration data and the <get> RPC to retrieve both configuration and operational data.

3. Issues with the Original Datastore Architecture

After some real world experience had been gained with the first version of NETCONF, NETCONF 1.1 was published as RFC 6241. A companion RFC, RFC 6244 "An Architecture for Network Management using NETCONF and YANG" was also published.

RFC 6244 identified an issue which arises when considering configuration and operational data. Configuration data, it defined, is the set of writable data that is required to transform a system from its initial default state into its current state. The RFC viewed operational state data as that which is obtained by the system at runtime and influences the system's behavior similar to configuration data. However, operational data can be transient and can be modified by interactions with internal components. For example, with interfaces on a device, some interfaces will be configured, but some interfaces, such as a default loopback interface, may be supplied by the device itself, and will exist independently of being configured.

Trying to combine the configuration and operational data in a YANG data model proved difficult. Various options were considered and a conclusion was reached that the only viable option was to distinctly model the configuration and operational values, where the configuration would indicate the desired (i.e. intended) value as specified by the user and the operational value would indicate the current value as it was observed on the device.

This resulted in YANG data models, at least those produced within the IETF, having multiple parallel lists, one for configuration data and one for operational data. For example, in RFC 7223, which specifies the YANG data models for interfaces, there are two containers, one called `container interfaces` which contains `list interface` for configuration information, and `container interface-state` which contains a `list interface` for operational information.

This solution has several problems. One is that there is duplication in the YANG data model which can cause redundant information. It also makes the two sets of information complicated since not all data may be duplicated. As a result, network managers may need to merge the two sets of information to form an overall view. Finally, the parallel sets of information are only a modeling convention and can't be enforced on YANG data model authors. This means that network managers can't depend on these parallel sets of information being present in all YANG data models.

In addition to the problem of maintaining a separation between configuration and operational data, some devices allow for pre-configuration and pre-provisioning. For example, you may want to pre-configure interfaces on the device prior to the planned addition of hardware, so that once the new line-card is inserted, the interfaces have already been configured. This raises the question, when examining the configuration of a device, of knowing which parts of the configuration have been applied and which are inactive due to missing hardware.

4. Network Management Datastore Architecture (NMDA)

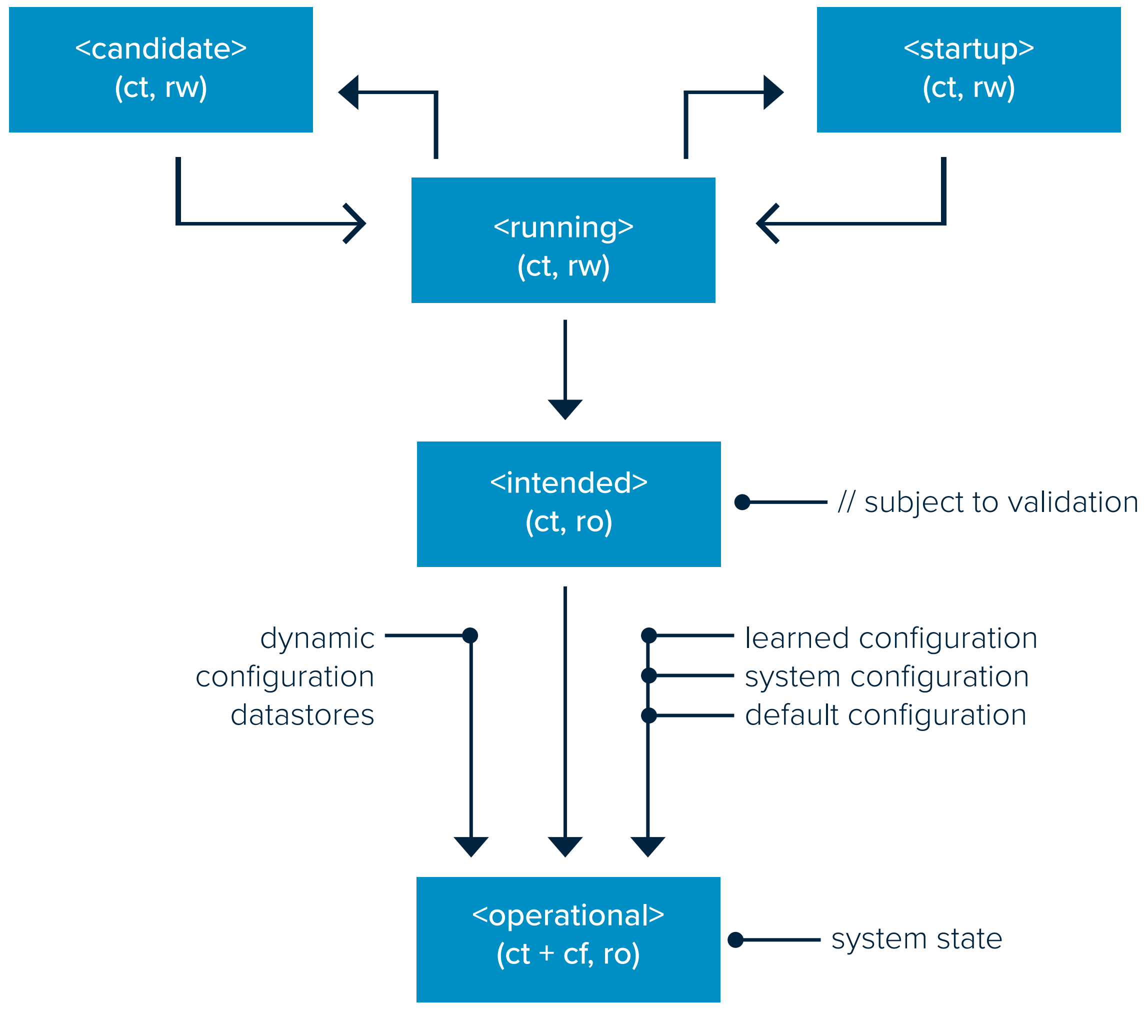
Network Management Datastore Architecture (NMDA), as specified in RFC 8342, attempts to solve these issues. NMDA builds on the original configuration datastore architecture of <startup>, <running>, and <candidate> and extends it with two additional datastores: <intended> and <operational>. Thus, NMDA takes the original datastore concept which applied only to configuration data and extends it to cover operational state data. In NMDA, <running>, <intended>, and <operational> are mandatory datastores, while <startup> and <candidate> remain optional.

The <running> datastore contains the complete current configuration of the device. The <running> datastore can be used in a read-write mode, or if the <candidate> datastore is supported, can be updated by writing through the <candidate> datastore.

The new <intended> datastore is a read-only configuration datastore that contains the configuration as it is intended to be used by the device. The difference between the <running> datastore and <intended> datastore is that some transformations may be performed on the <running> datastore, such as removing inactive configuration, to produce the <intended> datastore. Changes to the configuration of the device will still be made to <running>, whether directly or through the <candidate> datastore, and the <intended> datastore is a read-only representation of the <running> datastore once any transformations on the <running> datastore have been made.

The <operational> datastore is also read-only datastore and contains the operational state of the device. The schema for the <operational> datastore contains all config true and config false data. Retrieving from the <operational> datastore can return configuration data as well as operational data learned from the system, such as the default loopback interface from the previous section, and operational data such as counters and statistics. When retrieving from the <operational> datastore, the system has the choice to mark configuration data with an "origin" metadata annotation, which is a YANG identity, indicating where the data comes from. Some of the defined origins include "intended", "dynamic", and "system".

The figure below shows the relationships between the datastores in the Network Management Datastore Architecture:



Legend:
ct = config tru cf = config false
ro = read only rw = read-write

One consequence of NMDA is that YANG data models no longer need to keep parallel sets of information, i.e. one for configuration data and one for operational data. The IETF is in the process of updating YANG data models which have this structure. An example of this is the YANG data model for interfaces which was updated in RFC 8343.

5. Support for NMDA in NETCONF and RESTCONF

In order to support NMDA, a pair of new NETCONF RPCs are defined in RFC 8526. One of the new operations is <get-data>, which is similar to <get-config>, but it allows for a datastore to be specified as a parameter. In addition to the filtering capabilities of both sub-tree and XPath, there is a Boolean which can indicate whether only config true or config false data should be returned, as well as origin filters, which can make use of the "origin" metadata annotation in the <operational> datastore.

The other new operation is <edit-data>, which is similar to <edit-config>, but it allows for the datastore to be specified as a parameter. The <edit-data> operation can only be performed against a writable datastore.

In addition to the new operations, the <lock>, <unlock>, and <validate> operations were augmented with a new leaf in order to specify the datastore for the operation.

Enhancements to RESTCONF to support NMDA are specified in RFC 8527. The main additions are new datastore resources: {+restconf}/ds/<datastore>. With a few exceptions, protocol operations defined in the original RESTCONF specification, RFC 8040, for {+restconf}/data are available for the new datastore resources. There is also a "with-origin" query parameter that can be used when doing a GET operation on the <operational> datastore.

6. Implementation of NMDA in ConfD

In ConfD 7.3, support was added for NMDA, with the exception of the "origin" metadata attribute. Support for "origin" is planned to be added in ConfD 7.4.

There are no changes to the <startup>, <running>, and <candidate> datastores. The <intended> datastore is the <running> datastore after any transformations, where the only transformation from <running> to <intended> supported by ConfD is removing configuration items marked with the "inactive" metadata tag. For details on how to make use of the inactive metadata tag, please refer to the section "Meta-Data: inactive" in the ConfD User Guide chapter "Configuration Meta-Data".

For the <operational> datastore, ConfD will continue to work as before with your applications, where ConfD will read configuration data stored in CDB from CDB or from your data provider if the configuration data is stored outside of CDB. Config false data will be read from CDB if the operational data is stored in CDB or from your data provider.

A change that has been introduced is a new Tail-f YANG sub-annotation to tailf:callpoint called tailf:operational which has been added. When you add a tailf:callpoint with tailf:operational, this tells ConfD to call your data provider when the operation is against the <operational> datastore. This allows you, for example, to return system provided interfaces, such as the earlier example of a default loopback interface, when a <get-data> operation is performed on the <operational> datastore. Your data provider is only invoked for operations against the <operational> datastore, operations against other datastores will have the configuration data provided by CDB. For more details, see the section on “The Operational State Datastore” in the ConfD User Guide chapter “Operational Data”.

Another change that was introduced is to support YANG data models which might only be accessible in the <operational> datastore, for example, a YANG data model showing available hardware in the system. If you have a YANG data model which is only accessible in <operational>, then confdc supports a new option, “--datastore operational” which will indicate that the YANG data model should only be present in <operational>.

7. Summary

NMDA is an important new core standard for the world of NETCONF, RESTCONF, and YANG driven programmability. This application note has provided an introduction to NMDA by looking at why it exists, what it is, and how it is implemented in ConfD.

8. For More Information

For more information about ConfD, visit https://www.tail-f.com

RFC 4741 “NETCONF Configuration Protocol”: https://tools.ietf.org/html/rfc4741

RFC 6241 “Network Configuration Protocol (NETCONF 1.1)”:
https://tools.ietf.org/html/rfc6241

RFC 6244 “ An Architecture for Network Management Using NETCONF and YANG”:
https://tools.ietf.org/html/rfc6244

RFC 8342 “Network Management Datastore Architecture (NMDA)”:
https://tools.ietf.org/html/rfc8342

RFC 8526 “NETCONF Extensions to Support the Network Management Datastore Architecture”: https://tools.ietf.org/html/rfc8526

RFC 8527 “ RESTCONF Extensions to Support the Network Management Datastore Architecture”: https://tools.ietf.org/html/rfc8527 tail-f a Cisco company

www.tail-f.com
info@tail-f.com

Corporate Headquarters

Sveavagen 25
111 34 Stockholm
Sweden
+46 8 21 37 40