Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 8, 2009

P. Shafer
Juniper Networks

September 4, 2008

An Architecture for Network Management
draft-shafer-netmod-arch-00

Status of this Memo

Copyright Notice

Abstract

NETCONF and YANG are pieces of an ambitious plan to improve network management. NETCONF gives access to native capabilities of the devices within a network, defining methods for manipulating configuration databases, retrieving operational data, and invoking specific operations. YANG provides the means to define the content trafficked via NETCONF, both data and operations. Using both technologies, the standards modules can be defined to give interoperability and commonality to devices, while still allowing devices to express their unique capabilities.

This document describes how NETCONF and YANG help build network management applications that meet the needs of network services providers. An architecture is described which is friendly to both devices and applications, to vendors and standards bodies, to young and to old, to red states and to blue states. It's a startling vision, coming to networks near you starting August, 2009.

Table of Contents

1. Key Words

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 BCP 14, [RFC2119].

2. Introduction

Networks are increasing in complexity and capacity, as well as the density of the services deployed upon them. The drive for uptime, reliability, and predictable latency drives the need for automation.

The problems with network management are not simple. They are complex and intricate. But these problems must be solved for networks to meet the stability needs of existing services while incorporating new services in a world where the growth of the networks is exhausting the supply of qualified networking engineers. We need to move from a CLI world into a world of automation, but that automation must be robust and trustworthy.

This document presents an architecture based on NETCONF ([RFC4741]) and [YANG]. NETCONF and YANG address the problems of network management with flexibility and expressiveness. NETCONF allows any manner of configuration and operational data to be carried with few rules governing the data. YANG allows data models to be defined that are flexible and extensible in ways that allow the data to be cohesive and structured, but not rigid.

This approach allows the device to express its native capabilities in a way that is flexible and extensible. Evolution of devices and data models are permitted and managed.

2.1. NETCONF

NETCONF allows applications to use an XML-based RPC mechanism that leverages the simplicity and availability of high-quality XML parsers. XML gives a rich, flexible, hierarchical, standard representation of data that matches nearly perfectly the data needs of networking devices.

XML's hierarchical data representation allows complex networking data to be rendered in a natural way. For example, the following configuration places interfaces in OSPF areas. The <ospf> element contains a list of <area> elements, each of which contain a list of <interface> elements. The <name> element identifies the specific area or interface. Additional configuration for each area or interface appears directly inside the appropriate element.

```
<ospf xmlns="http://ns.ietf.org/netconf/ospf">

  <area>
    <name>0.0.0.0</name>

    <interface>
      <name>ge-0/0/0.0</name>
      <!-- The priority for this interface -->
      <priority>30</priority>
    </interface>

    <interface>
      <name>ge-0/0/1.0</name>
    </interface>
  </area>

  <area>
    <name>10.1.2.0</name>

    <interface>
      <name>ge-0/0/2.0</name>
    </interface>

    <interface>
      <name>ge-0/0/3.0</name>
    </interface>
  </area>
</ospf>
```

2.2. YANG

YANG is a data model language for NETCONF that allows the description of hierarchies of nodes and the constraints that exist amongst them. These data models are extensible in a manner that allows tight integration of standard data models and proprietary data models.

```
+--------------------------------------------------------------+
| Open Question                                                |
+--------------------------------------------------------------+
| More words?                                                  |
+--------------------------------------------------------------+
```

```
module ietf-ospf {
    namespace http://ns.ietf.org/netconf/ospf;
    prefix ospf;

    import network-types {
        prefix nett;
    }

    container ospf { // declare the top-level tag
        list area {
            key name;
            leaf name {
                type nett:area-id;
            }
            list interface {
                key name;
                leaf name {
                    type nett:interface-name;
                }
                leaf priority {
                    type uint {
                        range 1..255;
                    }
                }
            }
        }
    }
}
```

Open source tools for YANG are available on http://www.yang-central.org.

2.2.1. XML Namespaces

XML includes the concept of namespaces, allowing XML elements from different sources to be combined in the same hierarchy without risking collision. YANG modules define content for specific namespaces, but one module may augment the definition of another module, introducing elements from that module's namespace into the first module's hierarchy.

For example, if the above OSPF configuration were the standard, a vendor module may augment this with vendor-specific extensions.

```
module vendorx-ospf {
    namespace http://vendorx.example.com/ospf;
    prefix vendorx;

    import ietf-ospf {
        prefix ospf;
    }

    augment ospf:ospf/ospf:area/ospf:interfaces {
        leaf no-neighbor-down-notification {
            type empty;
            description "Don't inform other protocols about"
                      + " neighbor down events";
        }
    }
}
```

The <no-neighbor-down-notification> element is then placed in the vendorx namespace:

```
<protocols xmlns="http://ietf.org/netconf/protocols"
           xmlns:vendorx="http://vendorx.example.com/ospf">
  <ospf xmlns="http://ietf.org/netconf/ospf">

    <area>
      <name>0.0.0.0</name>

      <interface>
        <name>ge-0/0/0.0</name>
        <priority>30</priority>
        <vendorx:no-neighbor-down-notification/>
      </interface>

    </area>
  </ospf>
</protocols>
```

Extensions are seamlessly integrated with base modules, allowing them to be fetched, archived, loaded, and deleted within their natural hierarchy.

3. An Architecture for NETMOD

In the NETMOD architecture, each device vendor implements a set of data models in their devices. These models are either standard data models, defined in YANG modules published by a standards body, or proprietary data models, defined in YANG modules published by vendor.

Proprietary data models allow the vendor to accurately describe the content and behavior of their devices in explicit detail. Applications may take advantage of these specifics to give their users complete control over the device.

Standard data models define content that is independent of the vendor, allowing client applications to request specific behavior without concern for the vendor, product line, or installed software revision. The translation between the standard model and the device specific behavior is performed by the device, freeing the application from such concerns.

When a NETCONF session begins, the namespaces for all supported modules are announced as capabilities via the device's <hello> message. The device should also support the schema discovery mechanism [ref], enabling applications to discover the location from which the modules may be downloaded.

The schema discovery for standard YANG modules should list a common, standard location for these modules, assumably one set by the organization that defined the standard.

When an application connects with a device, it receives the list of capabilities supported by that device. The application may compare the set of capabilities announced by the device with the set of modules the application is aware of. Any new modules or new revisions of known modules may be downloaded as needed from the locations given via the schema discovery mechanism.

Once the application has access to the YANG modules, it may manipulate the device as a "YANG module browser", capable of parsing the elements sent from the device with an understanding of the organization of the data. The module describes the syntax of the data and constraints on that data, allowing the application to create data that abides by those constraints.

To have a real understanding of a module's content, the application may need to incorporate logic specific to that module. Semantic information contained in description statements is not machine readable, but module-specific custom work can be done to tailor the user interface to the particular semantic needs of the module.

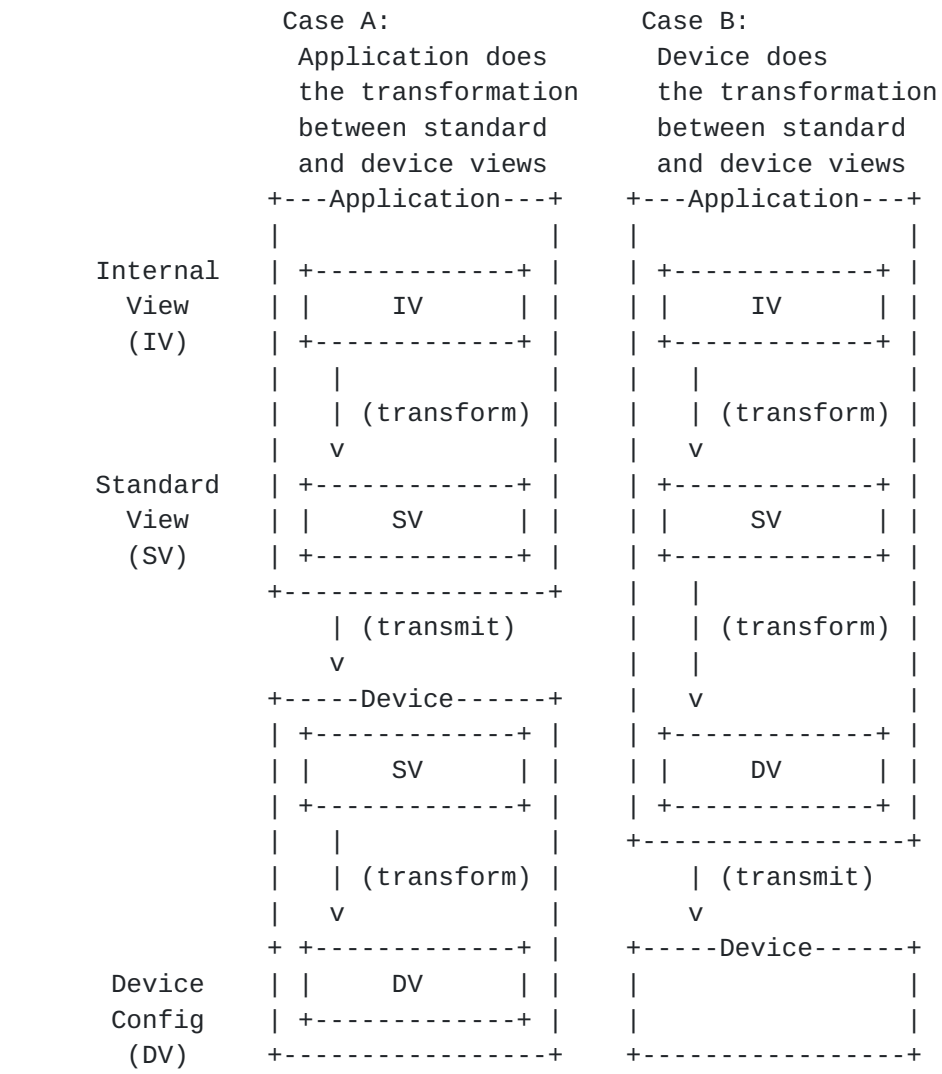
For example, a module could define the "location" of a device using longitude and latitude, and the application can use the "browser" style to display this data using input fields in a web form. Custom logic would be needed to take the value of these fields and place the device on a map of the world, and more logic would be needed to update the data values when the user drags the device from Dallas to Dulles.

If an application is meant to manage a specific problem, it may model the data internally in whatever form is most convenient to its organizational needs. When the application interacts with a device, it may choose one of two paths. If the device implements a standard module, the application may generate content for that standard by translating its internal form into the standard one.

If the device doesn't implement such a standard or no such standard exists, the application may use a transformation that is particular to that device's vendor, product model, hardware, or software. Depending on the application, this transformation may be provided by the application vendor, the device vendor, a third-party, or the provider.

For a popular application, the vendor may wish to provide this transformation to increase the uptake of their devices. For popular devices, the application may provide this transformation as a means of making the application useful in the maximum number of provider networks. For problem domains where the mapping from the application to the device is not straight-forward or requires tailoring to the specific provider or environment, the provider may wish to control this transformation. Additionally, other parties may make such transformations available via open source.

This gives two similar views of the world, as only ASCII Art can explain:



Open Question
Do we need this picture? Does it add anything worth adding?

Note that both cases may appear within a single application on an "as needed" basis. If the device announces the capability for the standard YANG module, the application may transmit to the device via NETCONF the content in the standard modules format. If the device does not announce the appropriate capability, the application may find a transformation that matches the device, perform the transformation on the standard data to produce device native configuration, and transmit via NETCONF that device configuration to the device.

In both cases, the key is the ability to discover the capabilities of

the specific device, download the YANG modules that support those capabilities, gain an understanding of those data models and their constraints, generate appropriate content, and transmit that content to the device.

4. YANG and Friends

The YANG data modeling language is the central piece of a group of related technologies. The YANG language itself, described in [ref], defines the syntax of the language and its statements, the meaning of those statements, and how to combine them to build the hierarchy of nodes that describe a data model.

That document also defines the "on the wire" XML content for NETCONF operations on data models defined in YANG modules. This includes the basic mapping between YANG data tree nodes and XML elements, as well as mechanisms used in <edit-config> content to manipulate that data, such as arranging the order of nodes within a list.

YANG is a fairly simple language, using a syntax that is regular and easily described, designed for human readability. But in some environments, incorporating a YANG parser may not be an acceptable option. For those scenarios, a XML grammar for YANG is defined in YIN (YANG Independent Notation) [ref]. YIN allows the use of XML parsers which are readily available in both open source and commercial versions. Conversion between YANG and YIN is direct, loss-less and reversible. YANG statements are converted to XML elements, preserving the structure and content of YANG, but enabling the use of "off the shelf" XML parsers rather than requiring the integration of a YANG parser. YIN maintains complete semantic equivalence with YANG.

Open Question
Should we note that YANG parsers are also available "off the shelf" also?

Difficult as it is to believe, there will be environments where tools are not available which offer native support. To support developers marooned in such environments, YANG offers the ability to translate modules into DSDL schemas, including RelaxNG and Schematron. With these schemas, existing tools may test that NETCONF operations and content are valid and well-formed before processing them. The complete standard mapping of YANG to DSDL is detailed in [ref], and is implemented in "off the shelf" open source and commercial software. Where no equivalent functionality is availabe in DSDL, annotations will be used to ensure the preservation of high-level semantics. Many DSDL engines offer plug-able functionality to allow enforcement of such constraints.

In addition, a standard type library for use by YANG is available in

[ref]. These types allow access to the vast wealth of the Sierra Madra, in true John-Wayne style.

4.1. Applicability

The data model in the YANG module yields value in five specific areas.

4.1.1. Device Developer

The YANG model tells the device developer what data is being modeled. The developer reads the YANG models, absorbs the zen of the model, and writes code that supports the model. The model describes the data hierarchy and associated constraints, and the description and reference material helps the developer understand how to transform the models view into the device native implementation.

4.1.2. Generic Content Support

The YANG model can be compiled into a YANG-based engine for either the client or server side. Incoming data can be validated, as can outgoing data. The complete configuration database may be validated in accordance with the constraints described in the data model.

Serializers and deserializers, for generating and receiving XML content, can be driven by the meta-data in the model. As data is received, the meta-data is consulted to ensure the incoming XML element is valid at the current spot in the hierarchy.

The YANG-based toolset can also generate support material for development environment environments, including structure or object definitions, for inclusion at compile time. But care must be taken for the extensibility of YANG definitions. Nodes in the data model may be augmented by other YANG modules, both standard and proprietary.

4.1.3. XML "over the wire" Definitions

The YANG module dictates the XML encoding sent "over the wire", though actual transmission should be encrypted so as not to appear as readable text on the physical media. The rules that define the encoding are fixed, so the YANG module can be used to ascertain whether a specific NETCONF payload is obeying the rules.

4.1.4. Application Developer

The YANG module tells the application developer what data can be modeled. Developers can inspect the modules and take one of three

distinct views. In this section, we will consider them and the impact of YANG on their design. In the real world, most applications are a mixture of these approaches.

4.1.4.1. Hard Coded

An application can be coded against the specific, well-known contents of YANG modules, implementing their organization, rules, and logic directly with explicit knowledge. For example, a script could be written to change the domain name of a set of devices using a standard YANG module that includes such a leaf node. This script takes the new domain name as an argument and insert it into a string containing the rest of the XML encoding as required by the YANG module. This content is then sent via NETCONF to the devices.

This type of application is useful for small, fixed problems where the cost and complexity of flexibility is overwhelmed by the ease of hard coding direct knowledge into the application.

4.1.5. Bottom Up

An application may take a generic, bottom up approach to configuration, concentrating on the device's data directly and treating that data in without specific understanding.

YANG modules may be used to drive the operation of the YANG equivalent of a "MIB Browser". Such an application manipulates the device's configuration data based on the data organization contained in the YANG module. For example, a GUI may present a straight-forward visualization where elements of the YANG hierarchy are depicted in a hierarchy of folders or GUI panels. Clicking on a line expands to the contents of the matching content.

This type of GUI can easily be built by generating XSLT stylesheets from the YANG data models. An XSLT engine can then be used to turn configuration data into a set of web pages.

The YANG modules allows the application to enforce the set of constraints it defines without understanding any specifics of the YANG module. The application need only understand the YANG and its encoding rules. This sort of browser has been available for MIBs, but will have better information at its disposal with YANG.

4.1.5.1. Top Down

In contrast to the generic approach, the top-down approach allows the application to take a view of the configuration data which is distinct from the standard and/or proprietary YANG modules. The

application is free to construct its own model for data organization and to present this model to the user. When the data needs to be transported to device, the application transforms the application data into device data. This transformation is under the control and maintenance of the application, allowing the transformation to be changed and updated without affecting the device.

5. Modeling Considerations

In developing good data models, there are many conflicting interests the data modeler must keep in mind. Modelers need to be aware of four types of behavior in modeled device:

- [strict compliance] behavior that follow the model completely
- [modeled deviations] behavior that follows within deviations allowed by the model
- [allowable deviations] behavior that falls outside the model, but can still be handled
- [unacceptable deviations] behavior that is not at all consistent with the model

Consider a data model that contains the number of hotdogs that can be consumed before puking on a roller coaster. A simple model can simply list this element as a state node of the data model. Compliant models report the value, allowing clients to reconsider either another hotdog or another ride on the roller coaster.

The model may opt for a more complex model which limits this node to only apply to people that ride roller coasters. If an implementation does not support the ability to ride roller coasters, the value is uninteresting and such implementation are not required to conjure up a meaningless value.

Once the model is published, an implementer may decide to make this node configurable, where the standard model describes it a state data. The implementation reports the value normally and may have an "out of band" mechanism for reporting that this node behaves in a different manner than the standard. Applications capable of discovering such behavior can make allowances, but applications that do not continue treating the implementation as if it were compliant.

Rarely, implementations may make decisions that prevent compliance with the standard. Such occasions are horrible, regrettable, nasty, and a pox on the face of standardization. But they remain a part of reality, and modelers and application writers ignore them at their own risk. An implementation that reports the hotdog limit as "cow" would be difficult to manage, but one must expect to encounter such "cow reporters" in the field.

Despite this, both client and server should view the YANG module as a contract, with both side agreeing to abide by the terms. The modeler should do their best to be explicit about the terms of such a

contract, lest some weasel successfully explains how "cow" is a valid value. But both client and server implementations should strive to faithfully and accurately implement the data model described in the YANG module.

6. Conclusion

[Wherein we tell you what an amazing job we've done]

We have done an amazing job.

Many standardization efforts result in a "design by committee"-style camel instead of a horse, but the YANG design committee and the NETMOD working group have used the implementation experience of their members to build a data modeling language for NETCONF that balances simplicity, flexibility, and extensibility in a harmony unequaled since the dawn of the claw hammer. The development of multiple YANG implementations has given us early feedback on technical issues and helped crisp the wording of many issues in the YANG specification.

While we are aware of the difficulties that network management has traditionally faced and the number of bodies that lie at the base of the mountain we are scaling, we are certain that these tools are a vast improvement on the previous generation, and feel assured that this will help us evolve the network management world in a timely and stable way to allow the evolution of new and powerful applications that will deliver automation to fulfill the needs of providers and their networks.

7. Security Considerations

Security is an important task, and YANG should be an integral part of access mechanism defined for NETCONF. Access and Authorization (A&A) should also be integrated with traditional on-device mechanisms, such as RADIUS and local passwords. The SSH transport for NETCONF provides such facilities. In addition, these on-device A&A mechanism can provide additional constraints for the operations of a client targeting a NETCONF server. The definitions and limits of such constraints are not part of the current effort and will be addressed by future work.

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4741] Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.

[YANG] Bjorklund, M., Ed., "YANG - A data modeling language for NETCONF", draft-ietf-netmod-yang-00 (work in progress).

Author's Address

Phil Shafer
Juniper Networks

Email: phil@juniper.net

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).