Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 9, 2017

J. White
D. Black
Dell EMC
J. Leung
Intel Corporation

March 9, 2017

YANG Baseline Switch Profile Background
draft-wbl-rtgwg-yang-ci-profile-bkgd-01

Abstract

A YANG device profile is primarily a group of YANG models that are appropriate for use with a particular class of device or in specific device roles. This document provides background and describes the rationale for a baseline data center switch device profile, e.g., for top-of-rack switches in data center converged infrastructure. This rationale is based on the reuse of YANG models by the DMTF Redfish standard for management of converged infrastructure, but the resulting YANG device profile is intended to be useable by any YANG-based network management approach or protocol, as opposed to being specific to Redfish.

Status of This Memo

Copyright Notice

1. Introduction

Disclaimer - this is a -00 draft, whose primary goal is to capture the rationale for use of YANG for Redfish network management and the desirability of a baseline data center network switch profile, including providing technical background on the Redfish standard and its approach to network management. The draft content is not polished, and the organization of the material is likely to change.

A YANG device profile is primarily a group of YANG models that are appropriate for use with a particular class of device or in specific device roles. This document provides background and describes the rationale for a baseline data center switch device profile, e.g., for top-of-rack switches in data center converged infrastructure. The rationale is based on the reuse of YANG models by the DMTF Redfish standard for management of converged infrastructure, but the resulting YANG device profile is intended to be useable by any YANG-based network management approach or protocol; it is not intended to be Redfish-specific.

In support of this rationale, this document provides background on how YANG is used in the Redfish management framework; the YANG modules are translated to Redfish schema definitions in order to enable reuse of the models are with Redfish-defined management protocols and related functionality.

2. Motivation

A common management framework with accompanying set of protocols and device models is desirable in systems management use cases. A good example of this is in a converged infrastructure deployment within a data center. Applications, servers, storage, appliances, and networking elements are assembled to create a combined coherent platform. For the networking components in such an environment, there are platform management elements that are common with other types of systems such as thermal monitoring, physical enclosure, fans, and power supplies, as well as networking specific management elements to control the forwarding and filtering of network packets or the networking services. The common elements should be accessed and managed in a single way across all systems within the deployment.

Management, orchestration, and control of such a system benefits from a single approach that enables unified tools sets and simplifies operations.

The networking specific configuration within the converged infrastructure environment only needs a subset of all the possible networking protocols and services giving the converged controller only the minimum spanning control surface in terms of the models it can access. A breakdown of the needs of such a switch result in about 5-10 YANG modules (see Appendix A). These 5-10 modules should lead to common YANG-based data center network switch management across vendors and products.

As a contrast, a full function edge router would need many more protocols and services along with full function virtualization resulting in the use of about 80 YANG modules.

2.1. Redfish Background

The DMTF (Distributed Management Task Force) Redfish standard is becoming the common standard for converged infrastructure (CI) management. Converged Infrastructure consists of rack-scale (partial or full-rack) integrated compute, network and storage infrastructure that is procured and deployed as rack scale systems.

Redfish data center storage management functionality has been extended in partnership with SNIA (Storage Networking Industry Association) resulting in the recently released Swordfish specification that extends Redfish for networked storage and storage network management. The authors hope to work with the IETF to extend and align Redfish network management with IETF’s YANG framework.

Redfish is a management standard using a data model representation inside of a RESTful interface. The model is expressed in terms of a standard, machine-readable schema, with the payload of the messages being expressed in JSON.

Because it is a model-based API, Redfish is capable of representing a variety of implementations via a consistent interface. It has mechanisms for managing data center resources, handling events, long lived tasks and discovery, as well.

In Redfish, every URL represents a resource. This could be a service, a collection, an entity or some other construct. But in RESTful terms, these URIs point to resources and Redfish clients interact with these resources. For example, the content of a resource, in JSON format, is returned when the Redfish client performs a HTTP GET.

OData is an OASIS standard for expressing the schema of a JSON document. OData allows a fuller description of the JSON document, than json-schema. The format of OData schema is specified in CSDL (Common Schema Definition Language).

OData also describes directives that can appear on the URI to control the contents of the HTTP response. In Redfish, these directives (i.e. $top and $skip) are stated as ’should’. Networking extension may change the requirement to ’shall’.

Redfish releases include both OData and json-schema schema. With json-schema, users can take advantage of its larger tool chain.

Additional information about OData can be found at the OData site [1].

Additional information about Redfish can be found at DMTF’s Redfish site [2]. Specifically, the Redfish White Paper [3] provides a good overview.

2.2. YANG and Redfish

Within the networking world, YANG has become the preferred management framework. YANG expresses each section of the overall model as a module containing a tree of nodes where each node is either a container node that builds the hierarchy or a leaf node containing data elements for the model. Redfish network management leverages YANG as the core model definition language to maintain consistency with other YANG based network management approaches. Using a common model structure results in equivalent data elements yielding the same data or content when accessed via different interfaces or protocols.

Redfish’s network management supports this consistency by reusing YANG modules as Redfish models for network functionality and services, mechanically translating those modules to the Redfish interface, based on HTTP, JSON, and OData.

The Redfish approach to network management enables definitions of a specific system views or profiles that are aligned with the configuration functionality required in a specific scenario or for a specific class of network devices. A particularly relevant example is a baseline switch model description with a minimum set of model elements; this is useful when configuring a switch within a larger converged infrastructure system. The model elements of the baseline switch should be the smallest set of models necessary to configure a data center switch in a converged infrastructure environment; the corresponding set of YANG modules would be a simple data center network device profile. A more complex network router might need tunnel models, overlay models, extensive QoS models, and other elements.

The top level resource structure of Redfish is show below.

```
./redfish/v1/Systems
./redfish/v1/Chassis
./redfish/v1/NetworkDevices
(other redfish resources)
```

Within this structure a network switch is viewed as a data center element for its common subsystems such as chassis, power, thermal, cooling, management access setup, etc while the network modeling is specified within the instances of the NetworkDevices[] collection. Each member of the NetworkDevices[] collection has implements its own set translated YANG modules. For consistency, the set of modules grouped under a NetworkDevice instance can follow one of multiple standard groupings expressed as a profile to manage different classes of equipment and satisfy different use cases. Two profile examples are the basic switch and virtualized edge router.

2.3. YANG mapping to Redfish

The notion of schema is different in Redfish and YANG.

In YANG, a schema is device specific. The user determines the YANG modules utilized by a system, and may consult a module library as part of doing so (e.g., RFC7895 [4]). The YANG schema is realized as a set of YANG modules, each with a prescribed node tree structure.

In Redfish, there is one schema that encompasses the entire namespace. In other words, Redfish has a global namespace for its schema, of which the device implements a subset. The Redfish schema is realized as resources accessed via a URI, so the namespace contains the information about which YANG modules are utilized. The OData directives $expand and $filter allows the client to discovery this directly from the parent namespace node above the modules.

That functionality obviates any Redfish need to use YANG module combination techniques such as YANG Schema-mount [5].

Despite these differences, the proposed profiles should be usable by both YANG based protocols (e.g., NETCONF, RESTCONF) and Redfish, as the core content of each profile is a set of YANG modules.

To allow Redfish to manage network devices, the YANG modules needs to be translated into OData CSDL (Common Schema Definition Language). The translation is specified in the YANG-to-Redfish Mapping

Specification [6]. The translation has the following characteristics:

- o Includes, imports, and augments, are compiled out to create a single consistent schema block constituting a particular instance of a NetworkDevice.

- o The YANG node tree layout is reflected in the URI layout

- o The individual YANG container nodes and list nodes are rendered as resources with the YANG tree hierarchy reflected as navigation properties.

Access to the YANG data model elements uses a Redfish JSON accessed via a provider on the URI target.

Leaf nodes representing common back end system "features or elements" return consistent data independent of node name and network device hierarchy.

The NetworkDevices[] collection allows

- o Multiple co-existing and consistent views onto a system.

  - * Horizontally extensible

  - * Vertical hierarchy allowing for control interface delegation

- o This is similar to a "view class" or facade approach in software.

2.4. Example Mapping

The following shows the resource which results from mapping RFC7223 (ietf_interface module) to the Redfish schema. Below is a fragment of the data model from the RFC.

```
+--rw interfaces
| +--rw interface* [name]
| +--rw name string
| +--rw description? string
| +--rw type identityref
| +--rw enabled? boolean
| +--rw link-up-down-trap-enable? enumeration
+--ro interfaces-state
+--ro interface* [name]
+--ro name string
+--ro type identityref
+--ro admin-status enumeration
```

The translation to Redfish CSDL is performed using the RFC’s YANG code. The translation will generate the CSDL files for the ietf_interfaces resource and each YANG container. The path to these resources mirror the above data model.

```
./redfish/v1/NetworkDevices/Switch1
./redfish/v1/NetworkDevices/Switch1/ietf_interfaces
./redfish/v1/NetworkDevices/Switch1/ietf_interfaces/interfaces
./redfish/v1/NetworkDevices/Switch1/ietf_interfaces/interfaces/ethernet1
./redfish/v1/NetworkDevices/Switch1/ietf_interfaces/interfaces_state
...
```

A HTTP GET of the "ethernet1" singleton resource will return the following JSON document. Note that each property from the above data model is present in the resource.

```
{
    "Id": "ethernet1",
    "Name": "ethernet1",
    "Description": "Ethernet interface on slot 1",
    "type": "iana_if_type:ethernetCsmacd",
    "enabled": "true",
    "link_up_down_trap_enable": "true"

    "@odata.context":
        "/redfish/v1/$metadata#ietf_interfaces.interfaces.interface.
            interface",
    "@odata.type": "#interface.v1_0_0.interfaces",
    "@odata.id":
        "/redfish/v1/NetworkDevices/Switch1/ietf_interfaces
            /interfaces/ethernet1"
}
```

The three properties at the end of the JSON document are OData annotations.

3. Security Considerations

Redfish also improves security control since there is a single point of management contact for a device to control all of its functions.

(Additional security discussion will be provided later.)

4. Appendix A

YANG models needed to managed a network switch:

o RFC7223 (Interfaces)

- RFC7224 (IANA)
- RFC7277 (IPv4, IPv6)
- RFC7317 (System Identification, Time-Date, NTP)
- VLANs
- ACLs
- Syslog

5. Appendix B

The following describes how the Redfish NetworkDevices[] collection resource allows multiple co-existing and consistent views onto a system.

As an example, a router could have the following:

```
//redfish.example.net/redfish/v1/NetworkDevices/masterRouter
//redfish.example.net/redfish/v1/NetworkDevices/vrf1
//redfish.example.net/redfish/v1/NetworkDevices/vrf2
```

In this example, masterRouter represents the complete system with all interfaces, all tables, all system level configuration, and a model structure for assigning resources to virtual instances. The resources, vrf1 and vrf2, represent a particular partitioning of the system to create virtual router instances each assigned a subset of the total resource pool.

The above structure has similarities with that expressed by the device model from the following references:

- https://tools.ietf.org/html/draft-ietf-rtgwg-device-model-01
- https://tools.ietf.org/html/draft-ietf-rtgwg-ni-model-01
- https://tools.ietf.org/html/draft-ietf-rtgwg-lne-model-01
- https://tools.ietf.org/html/draft-ietf-netmod-schema-mount-03

In these references a Network Device contains Logical Network Elements which, in turn, contain Network Instances. From the device model reference, the Network Device represents the system as a whole. The Logical Network Element represents a partition of a physical system. The Logical Network Element represents a VRF or VSI (virtual switching instance).

The Redfish NetworkDevices collection resource would map this modeling approach by using an element of the collection for the Network Device and one for each of the Logical Network Elements and Network Instances. These collection elements would add references at the NetworkDevices element level to map the containment of of the device model. The overall ./redfish/v1/ root maps to the Routing Area Network Device.

6. Appendix C

The following is the ietf_interfaces.interfaces.interface_v1.xml CSDL metadata file, which is referenced in @odata.context annotation in the example mapping. The entity type referenced in the @odata.type annotation is in the second Namespace.

When mapping YANG code to CDSL, values are mapped to existing OData core properties, when possible. Otherwise, new annotations are defined in RedfishYangExtensions.xml. This file is referenced at the beginning of the document.

```
<?xml version="1.0" encoding="UTF-8"?>
<edmx:Edmx xmlns:edmx="http://docs.oasis-open.org/odata/ns/edmx"
        Version="4.0">
    <Edmx:Reference
        Uri="http://docs.oasis-open.org/odata/odata/v4.0/cs01/
            vocabularies/Org.OData.Core.V1.xml">
        <Edmx:Include Alias="Odata" Namespace="Org.OData.Core.V1"/>
    </Edmx:Reference>
    <Edmx:Reference
        Uri="http://docs.oasis-open.org/odata/odata/v4.0/cs01/
            vocabularies/Org.OData.Capabilities.V1.xml">
        <Edmx:Include Alias="Odata"
            Namespace="Org.OData.Capabilities.V1"/>
    </Edmx:Reference>
    <Edmx:Reference
        Uri="http://redfish.dmtf.org/schemas/v1/
            RedfishYangExtensions.xml">
        <Edmx:Include Alias="Redfish.Yang"
            Namespace="Redfish.Yang"/>
    </Edmx:Reference>

    <Edmx:DataServices>

    <Schema Namespace="interface"
            xmlns="http://docs.oasis-open.org/odata/ns/edm" >
        <EntityType Name="interface"
                BaseType="Resource.v1_0_0.Resource">
            <Annotation Term="OData.Description"
```

```
                    String="<manual input>." />
                <Annotation Term="OData.AdditionalProperties"
                    Bool="False"/>
            </EntityType>
        </Schema>

        <Schema Namespace="interface.v1_0_0"
                xmlns="http://docs.oasis-open.org/odata/ns/edm" >
            <EntityType Name="interface" BaseType=
                    "ietf_interfaces.interfaces.interface.interface" >
                <Annotation Term="OData.Description"
                    String="<manual input>." />
                <Annotation Term="OData.AdditionalProperties"
                    Bool="False"/>
                <Annotation Term="Redfish.Yang.NodeType"
                    EnumMember="Redfish.Yang.NodeTypes/list" />
                <Annotation Term="Redfish.Yang.key"
                    String=" the yang key string"/>
                <Key>
                    <PropertyRef Name="name"/>
                </Key>
                <Property Name="name" Type="Edm:String">
                    <Annotation Term="OData.Description"
                        String="..." />
                    <Annotation Term="OData.Permissions"
                        EnumMember="OData.Permissions/Read"/>
                    <Annotation Term="Redfish.Yang.NodeType"
                        EnumMember="Redfish.Yang.NodeTypes/leaf"/>
                    <Annotation Term="Redfish.Yang.YangType"
                        String="string" />
                </Property>
                <Property Name="description" Type="Edm:String">
                    <Annotation Term="OData.Description"
                        String="..." />
                    <Annotation Term="OData.Permissions"
                        EnumMember="OData.Permissions/Read"/>
                    <Annotation Term="Redfish.Yang.NodeType"
                        EnumMember="Redfish.Yang.NodeTypes/leaf" />
                    <Annotation Term="Redfish.Yang.YangType"
                        String="string" />
                    <Annotation Term="Redfish.Yang.reference"
                        String="RFC 2863: The Interfaces Group..." />
                </Property>
                <Property Name="type" Type="Edm:String">
                    <Annotation Term="OData.Description" String="..."/>
                    <Annotation Term="Redfish.Yang.NodeType"
                        EnumMember="Redfish.Yang.NodeTypes/leaf" />
                    <Annotation Term="Redfish.Yang.YangType"
```

```
                         String="identityref"/>
                     <Annotation Term="Redfish.Yang.base"
                         String="interface-type"/>
                     <Annotation Term="Redfish.Yang.mandatory"
                         EnumMember="Redfish.Yang.Mandatory/true"/>
                     <Annotation Term="Redfish.Yang.reference"
                         String="RFC 2863: The Interfaces Group..." />
                 </Property>
                 <Property DefaultValue="true" Name="enabled"
                         Type="Edm:Boolean">
                     <Annotation Term="OData.Description"
                         String="This leaf contains..." />
                     <Annotation Term="Redfish.Yang.NodeType"
                         EnumMember="Redfish.Yang.NodeTypes/leaf" />
                     <Annotation Term="Redfish.Yang.YangType"
                         String="boolean"/>
                     <Annotation Term="Redfish.Yang.reference"
                         String="RFC 2863: The Interfaces..."/>
                 </Property>
                 <Property Name="link_up_down_trap_enable"
                         Type="Edm:Enumeration">
                     <Annotation Term="OData.Description"
                         String="Controls whether..."/>
                     <Annotation Term="Redfish.Yang.NodeType"
                         EnumMember="Redfish.Yang.NodeTypes/leaf"/>
                     <Annotation Term="Redfish.Yang.YangType"
                         String="enumeration"/>
                     <Annotation Term="Redfish.Yang.if_feature"
                         String="if-mib"/>
                     <Annotation Term="Redfish.Yang.reference"
                         String="RFC 2863: The Interfaces..." />
                     <EnumType
                             Name="link_up_down_trap_enableEnumeration">
                         <Member Name="enabled" Value="1">
                             <Annotation Term="Redfish.Yang.enum"
                                 String="enabled"/>
                         </Member>
                         <Member Name="disabled" Value="2">
                             <Annotation Term="Redfish.Yang.enum"
                                 String="disabled"/>
                         </Member>
                     </EnumType>
                 </Property>
             </EntityType>
         </Schema>

         </Edmx:DataServices>
```

```
     </edmx:Edmx>
```

7. Appendix D

The following is the IETF YANG source XML from RFC7223 used for the example mapping.

```
   <CODE BEGINS> file "ietf-interfaces@2014-05-08.yang"
   module ietf-interfaces {
       namespace "urn:ietf:params:xml:ns:yang:ietf-interfaces";
       prefix if;
       import ietf-yang-types {
           prefix yang;
       }
       organization
           "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
       . . .
```

After the typedef, identity, and feature statements, the data model is defined. Below is the fragment that becomes ietf_interfaces.interfaces.interface_v1.xml.

```
       /*
        * Configuration data nodes
        */
       container interfaces {
           description
               "Interface configuration parameters.";
           list interface {
               key "name";
               description
                   "The list of configured interfaces...";
           leaf name {
               type string;
               description
                   "The name of the interface...";
           }
           leaf description {
               type string;
               description
                   "A textual description of the interface...";
               reference
                   "RFC 2863: The Interfaces Group MIB - ifAlias";
           }
           leaf type {
               type identityref {
                   base interface-type;
               }
               mandatory true;
               description
                   "The type of the interface...";
               reference
                   "RFC 2863: The Interfaces Group MIB - ifType";
           }
           leaf enabled {
               type boolean;
               default "true";
               description
                   "This leaf contains the configured,...";
               reference
                   "RFC 2863: The Interfaces Group MIB - ifAdminStatus";

           leaf link-up-down-trap-enable {
               if-feature if-mib;
               type enumeration {
                   enum enabled {
                       value 1;
                   }
                   enum disabled {
                       value 2;
                   }
               }
               description
                   "Controls whether linkUp/linkDown SNMP...";
               reference
                   "RFC 2863: The Interfaces Group MIB -
                       ifLinkUpDownTrapEnable";
           }
       }
       . . .
```

8. References

8.1. Normative References

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

8.2. URIs

[1] http://odata.org

[2] http://dmtf.org/redfish

[3] http://www.dmtf.org/sites/default/files/standards/documents/DSP2044_1.0.0.pdf

[4] http://www.rfc-editor.org/info/rfc7895

[5] https://tools.ietf.org/html/draft-ietf-netmod-schema-mount-03

[6] http://www.dmtf.org/sites/default/files/standards/documents/
DSP0271_0.5.6.pdf

Authors' Addresses

Joseph White
Dell EMC

Email: joseph.l.white@dell.com

David Black
Dell EMC

Email: david.black@dell.com

John Leung
Intel Corporation

Email: john.leung@intel.com