VPN Service Management YANG Data Model  
draft-zaalouk-supap-vpn-service-management-model-01

Abstract

Currently new services create new opportunities for both network providers and service providers. Shared Unified Policy Automation (SUPA) was proposed to develop a model that abstracts network resources and services and a methodology by which the management and monitoring of network services can be done using standardized policy rules. This document defines a VPN service management yang data model and gives an example for DDC use case.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html
Currently new services bring new challenges and opportunities for both network providers and service providers. Meanwhile, legacy services such as VPN [RFC4110] also need specialized management and controlling capability from the network management systems to improve the experiences for fast deployment and dynamic configuration.

Shared Unified Policy Automation (SUPA) [SUPA-problem-statement] [SUPA-framework] was proposed to introduce the concepts of multi-level and multi-technology network abstractions to address the
The current separation between development and deployment operations. The first example that SUPA will focus on will be VPN management.

This document introduces YANG [RFC6020] [RFC6021] data models for SUPA configuration. Such models can facilitate the standardization for the interface of SUPA, as they are compatible to a variety of protocols such as NETCONF [RFC6241] and [RESTCONF]. Please note that in the context of SUPA, the term "application" refers to a operational and management applications employed, and possibly implemented, by an operator. The first configuration model is based on the first example - VPN management.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [RFC2119] significance.

3. Network Configuration Modules

In this section, several specific network configuration models are described based on a set of specific network services and the framework of SUPA [SUPA-framework].

3.1. L3VPN Service Module

A Layer 3 Virtual Private Network (L3VPN) interconnects sets of hosts and routers based on Layer 3 addresses and forwarding. L3VPN can be based on MPLS or IP technologies. L3VPN is a PE-based VPN managed by operators. L3VPN is widely used in carrier metro networks to provide VPN service for enterprise users.

A L3VPN model is a collection of L3VPN instances. A L3VPN instance contains a set of access interfaces to network devices as well as other attributes, such as routing protocol, address family, topology, and so on.
To configure a L3VPN instance, the administrator needs to specify which port(s) of a network device belongs to a L3VPN instance. Those ports and network device information can be derived from a network topology model in a network management system. The administrator also needs to specify what routing protocol needs to be configured for a L3VPN instance.

The following describes the information model for L3VPN, based on which programmers can develop applications to configure L3VPN instances.
module: ietf-supa-l3vpn
   +--rw l3vpn-Instance* [instance-name]
      +--rw instance-name       string
      +--rw servic-type?         enumeration
      +--rw address-family-type? enumeration
      +--rw access-interface-id* [access-interface-id]
         +--rw access-interface-id string
         +--rw access-interface-address string
         +--rw ip-address-mask-length uint8
         +--rw role                enumeration
         +--rw user-name           string
         +--rw user-password       string
         +--rw physical-node-id    string
         +--rw physical-access-interface string
         +--rw protocol
            +--rw protocol-type?   enumeration
            +--rw igp-attribute
            |   +--rw protocol-id?   uint32
            +--rw bgp-attribute
            |   +--rw remote-as-number? string
            +--rw remote-peer-address string

3.1.1. L3VPN YANG Model

<CODE BEGINS>
module ietf-supa-l3vpn {
   // replace with IANA namespace when assigned
   prefix l3vpn;
   organization "IETF";
   contact
   *Editor: Dacheng Zhang
dacheng.zdc@alibaba-inc.com
Adel Zaalouk
adel.ietf@gmail.com
Kostas Pentikousis
k.pentikousis@eict.de
*;

description
*This YANG module defines a component that describing
the ddc service model for creating and optimizing
tenant’s DC (data center) services that are deployed
in multiple data centers.*

Terms and Acronyms

L3VPN: Layer 3 Virtual Private Network

revision 2015-02-04 {

description

"Initial revision.";
reference "RFC4364, RFC7277";
}

list l3vpn-Instance {

key "instance-name";

max-elements "65535"; //to be discussed

description "Indicates the name of the VPN instance created.";

leaf instance-name {


type string {

length "1..64";

pattern "([^?]*)";

}
mandatory true;

description "L3VPN instance name.";
}

leaf servic-type {


type enumeration {

enum full-mesh {

value "0";

description "full-mesh";

}

description "full-mesh";

description "Topology type.";

}

default "full-mesh";

leaf address-family-type{

type enumeration {

enum ipv4uni {

value "0";

description "ipv4 unicast";

}

default "ipv4 unicast";

enum ipv6uni {

value "1";
  description "ipv6 unicast";
}
}
default "ipv4uni";
description "Address family type: IPv4 or IPv6.";
}

list access-interface-id {
  key "access-interface-id";
  max-elements "65535";
  description "Access interface ID."
}

leaf access-interface-id {
  type string {
    length "1..64";
    pattern "([^?]+)";
  }
  mandatory true;
  description "Access interface ID."
}

leaf access-interface-address {
  type string {
    pattern "([^?]+)"
  }
  mandatory true;
  description "Access interface address, IPv4 or IPv6.
}

leaf ip-address-mask-length{
  type uint8 {
    range "0..128";
  }
  mandatory true;
  description "IP address mask length."
}

leaf role {
  type enumeration {
    enum edge-if {
      value "0";
      description "edge interface";
    }
    enum center-if {
      value "1";
      description "center interface";
    }
  }
}
mandatory true;
description
"center-if is only available in hub-spoke mode;
center-if is the interface in hub node."
}

leaf user-name {
type string {
   length "1..64";
   pattern "([^?]*")";
}
mandatory true;
description "User name for this access interface.";
}

leaf user-password {
type string {
   length "1..64";
   pattern "([^?]*")";
}
mandatory true;
description "User password for the access interface.";
}

leaf physical-node-id {
type string {
   length "1..64";
   pattern "([^?]*")";
}
mandatory true;
description "Physical node ID.";
}

leaf physical-access-interface {
type string {
   length "1..64";
   pattern "([^?]*")";
}
mandatory true;
description "Physical access interface.";
}

container protocol {
description ".";}
leaf protocol-type {
  type enumeration {
    enum bgp {
      value "0";
      description "bgp";
    }
    enum ospf {
      value "1";
      description "ospf";
    }
    enum isis {
      value "2";
      description "isis";
    }
    default "ospf";
    description "Protocol type.";
  }
}

container igp-attribute {
  description ".";

  leaf protocol-id {
    type uint32 {
      default "0";
      description "Valid only when protocol is IGP;
      it can be AS number.";
    }
  }
}

container bgp-attribute {
  description ".";

  leaf remote-as-number {
    type string {
      length "1..11";
    }
    default "0";
    description "Valid only when protocol is BGP.";
  }

  leaf remote-peer-address {
    type string {
    }
    mandatory true;
    description "Valid only when protocol is BGP.";
  }
}
3.2. Module for DDC services

The following describes SUPA VPN management model designed for DDC services use case [SUPA-DDC]. [SUPA-DDC] took a large-scale Internet Data Center (IDC) operator as an example to describe what SUPA needs to do including DDC service initiation, VPN-based connectivity initiation, optimize traffic route, traffic adjustment and monitor.

Module "ietf-supadcc" defines generic VPN management aspects which are common to all DDC services use case regardless of their type of vendor. In effect, the module can be viewed as providing a generic VPN management for DDC services.
module: ietf-supa-ddc

```yang
++--rw createDdcServices
    +--rw ddcService* [tenantName]
        +--rw tenantName         string
        +--rw dcName*            string
        +--rw tenantNetworkId*   string
        +--rw connectionType?    enumeration

++--rw createVpnInstancesforDdc
    +--rw vpnInstance* [vpnName]
        +--rw vpnName                  string
        +--rw vlanId?                  uint16
        +--rw dataCenterInformation* [dcName]
            |  +--rw dcName           string
            +--rw interfaceName?   string
        +--rw vpnType?                 enumeration
        +--rw bandWidth?               uint32
        +--rw latency?                 uint32

++--rw optimizeTrafficServices
    +--rw optimizeTrafficService* [vpnName]
        +--rw vpnName               string
        +--rw vpnType?              enumeration
        +--rw bandWidth?            uint32
        +--rw latency?              uint32
```

3.2.1. Model for DDC services

```yang
<CODE BEGINS>
module ietf-supa-ddc {
    // replace with IANA namespace when assigned
    prefix ddc;

    organization "IETF";
    contact
        "Editor: Ying Cheng
         chengying10@chinaunicom.cn";

    description
        "This YANG module defines a component that describing
        the ddc service model for creating and optimizing
        tenant’s DC (data center) services that are deployed
        in multiple data centers.

    Terms and Acronyms
    DDC: Distributed Data Center
    L2VPN: Layer 2 Virtual Private Network
```

container createDdcServices {
  description
   "Management system/ application requires controller to
   create tenant’s network that are deployed in multiple
data centers. The controller(s) is/are told the following
data: name of data centers that the tenant’s service are
   deployed in, connected method between data centers for
   the tenant (e.g. L2VPN, L3VPN, Native IP, etc.), name
   of tenant, ID of networks that belong to the tenant";
list ddcService {
  key "tenantName";
  description
   "Overall ddc operational data, including the names of data
center, the connection method between data centers, name
   of tenant, ID of networks that belong to the tenants";
leaf tenantName {
  type string;
  mandatory true;
  description
   "Indicates the name of the tenant that the ddc service
   is created for";
}
leaf-list dcName {
  type string;
  description
   "List of the names of data center that the tenant’s
   service is deployed in.";
}
leaf-list tenantNetworkId {
  type string;
  description
   "List of the tenant networks in different data centers.
   These networks should be integrated into the tenant’s
   virtual data center";
}
leaf connectionType {
  type enumeration {
    enum L2VPN {
      value 0;
    }
}
description "L2VPN";

} enum L3VPN {
    value 1;
    description "L3VPN";
}

enum nativeIPv4 {
    value 2;
    description "L4VPN";
}

enum nativeIPv6 {
    value 3;
    description "nativeIPv6";
}

description
"Indicates the connection method between data centers
that the tenant service is deployed in. The connection
type may be VPN (L2VPN or L3VPN) or Native IP (IPv4 or
IPv6)";

}

container createVpnInstancesforDdc {
    description
    "Management system/ application requires controller to
create VPN for a tenant between data centers. VPN name,
tenant VLAN ID, VPN sites and interfaces, VPN type,
bandwidth requirement and latency requirement should be
told to controller";
    list vpnInstance {
        key "vpnName";
        description
        "Overall VPN operational data, including the name of VPN,
        the VLAN ID of tenant, the sites information of the VPN,
        the interface names of VPN endpoints, the type of VPN,
        the bandwidth and latency requirements of VPN";
        leaf vpnName {
            type string;
            mandatory true;
            description
            "Indicates the name of the VPN instance";
        }
        leaf vlanId {
            type uint16 {
                range "1 .. 4094";
            }
        }
    }
}

list dataCenterInformation {
  key "dcName";
  leaf dcName {
    type string;
    description
      "List of the names of data center that the tenant’s service is deployed in.";
  }
  leaf interfaceName {
    type string;
    description
      "Indicates a set of access interface names of the network device that the data centers (deployment of tenant’s service) are connected to.";
  }
  description
    "List of data center information including the names of data center and a set of access interface names of the network device";
}
leaf vpnType {
  type enumeration {
    enum L2VPN {
      value 0;
      description "L2VPN";
    }
    enum L3VPN {
      value 1;
      description "L3VPN";
    }
  }
  description
    "Indicates the type of VPN instance that is created for tenant. It can be L2VPN or L3VPN";
}
leaf bandWidth {
  type uint32;
  description
    "Indicates the bandwidth requirement of the VPN instance that is created for tenant.";
}
leaf latency {
  type uint32;
description
"Indicates the latency requirement of the VPN instance
that is created for tenant."
;
}
}

container optimizeTrafficServices {

description
"Management system/ application requires controller to
adjust the bandwidth of VPN to optimize the traffic when
the bandwidth utilization is below or over certain
threshold. vpn name, vpn type and adjusted bandwidth
should be told to controller."
;
list optimizeTrafficService {

type string;
mandatory true;

leaf vpnName {

leaf vpnType {

leaf bandWidth {

type uint32;

description
"Indicates the type of VPN instance that needs to be
adjusted. L2VPN or L3VPN";
}

leaf latency {

type string;

description
"Indicates the latency requirement of the VPN instance
that is created for tenant."
}

"Indicates the bandwidth requirement of the VPN instance that is created for tenant."
}
leaf latency {
  type uint32;
  description
    "Indicates the latency requirement of the VPN instance that is created for tenant.";
}

4. Security Considerations

TBD

5. IANA Considerations

This document has no actions for IANA.

6. Acknowledgments

This document has benefited from reviews, suggestions, comments and proposed text provided by the following members, listed in alphabetical order: Feng Dong, Jing Huang, Junru Lin, Felix Lu, Wu Nan, Juergen Schoenwaelder, Yiyong Zha, and Cathy Zhou.

Will Liu contributed to an early version of this draft.

7. References

7.1. Normative References

7.2. Informative References


[SUPA-DDC] Y. Cheng, and JF. Tremblay, "Use Cases for Distributed Data Center Applications in SUPA", IETF Internet draft, draft-cheng-supa-ddc-use-cases, January 2015.

Authors’ Addresses

Dacheng Zhang (Editor)
Alibaba
Chaoyang Dist
Beijing  100000
P.R. China
dacheng.zdc@alibaba-inc.com

Adel Zaalouk (Editor)
EICT GmbH
Torgauer Strasse 12-15
Berlin  10829
Germany
Email: adel.ietf@gmail.com

Kostas Pentikousis
EICT GmbH
Torgauer Strasse 12-15
Berlin  10829
Germany
Email: k.pentikousis@eict.de

Ying Cheng
China Unicom
P.R. China

Email: chengying10@chinaunicom.cn