Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 7, 2016

R. Shakir
Jive Communications

November 4, 2015

A Data Model for Network Instances in Service Provider Networks
draft-openconfig-rtgwg-network-instance-01

Abstract

This document defines a YANG data model for configuring and managing forwarding instances on a network device - focused primarily on deployments in service provider networks. The model is vendor-neutral and based on operational requirements expressed by operators. A 'network instance' is defined to be a discrete forwarding table on a network element, which contains Layer 3 and/or Layer 2 forwarding entries. Each network instance may instantiate its own sets of protocols which control installation of forwarding entries into one or more tables (which may be Layer 2 FIBs, or Layer 3 RIBs).

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 May 7, 2016.

Copyright Notice

Copyright (c) 2015 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.

1. Introduction

Within service provider networks, it is typical for a network element to maintain multiple forwarding tables, allowing the network to support multiple topologies. For instance, a service provider may offer Layer 2 or Layer 3 VPN services to its customers - which requires a forwarding table per customer to be maintained on a network element. In other cases, multiple L3 services may be deployed to allow services such as Internet connectivity and private topologies to be maintained on a device.

This YANG [RFC6020] data model defines a construct which can be used to configure and retrieve operational state related to such instances.

1.1. Goals and Approach

The concept of maintaining multiple Layer 2 virtual switch instances (VSIs) and/or multiple Layer 3 virtual routing and forwarding (VRF) tables is common on network elements that are deployed in service provider networks. However, within such networks there are requirements for a network instance to run both Layer 2 and Layer 3 routing protocols, or contain routed interfaces (e.g., a VPLS service where there is a Layer 3 interface within it, for subscriber termination or to act as a default gateway). For this reason, the intention of the model presented here is to define a generic construct which allows isolation of a forwarding table which may contain Layer 3 RIB entries, Layer 2 FIB entries, or a combination both. This differs to the approach taken in other models, which tend to focus on solely L2 VSIs or L3 VRFs. An instance within the model is referred to as network instance - and may be configured to support L3 RIB entries only (i.e., be functionally equivalent to a VRF), L2 FIB entries only (i.e., act as a VSI) or support a combination of both.

The model presented in this document is explicitly biased towards deployments on service provider network equipment, as discussed between members of the OpenConfig project.

2. Model overview

```
module: openconfig-network-instance
   +--rw network-instances
      +--rw network-instance* [name]
         +--rw name                        -> ../config/name
         +--rw config
            ...
         +--ro state
            ...
         +--rw inter-instance-policies
         |  +--rw apply-policy
            ...
         +--rw table-connections
         |  +--rw table-connection* [src-table dst-table]
         |     +--rw src-table       -> ../config/src-table
         |     +--rw dst-table       -> ../config/dst-table
                ...
         |     +--rw apply-policy
                ...
         +--rw tables
         |  +--rw table* [table-name]
                ...
         +--rw interfaces
                ....
         +--rw connection-points
         |  +--rw connection-point* [connection-point-id]
              ...
         +--rw protocols
            +--rw protocol* [identifier name]
                ...
               +--rw static
                ...
               +--rw aggregate
                ...
```

The key elements of the network instance model shown in the above outline tree view are:

- Type of network instance and key configuration parameters that relate to it (e.g., configuration of a route distinguisher for L3 routing instance).

- Policies which define how the instance exchanges routes with other instances - for example, allowing forwarding entries to be leaked between a global and private network instance.

- The individual tables that are maintained by a network instance. This caters for the approach whereby an implementation maintains per-protocol tables - and it is possible to examine these tables separately. The choice to implement this table definition is an open issue - as discussed in Section 3.1. Along with this table definition, a set of policies determining how entries are leaked between the tables is defined.

- Connections between the different tables that are maintained by the network instance. Where protocols populate individual tables, such connections facilitate leaking of routes between individual protocols (by means of populating entries from one into the table of the other).

- The interfaces associated with a certain network instance - the current implementation assumption (as demonstrated in the openconfig-interfaces models) is that configuration parameters that relate directly to the interface (e.g., IP addressing) are configured directly under the interface construct. This requires that a network instance must have a means to be able to associate itself with an interface.

- A logical grouping of local or remote interfaces into 'connection points' utilised by a number of Layer 2 forwarding approaches (e.g., redundancy for attachment points for PWE or L2VPN services). This allows interfaces which are associated with the network instance to be associated with one another, and hence referenced as a group. This construct can be used to represent redundant interfaces for a P2P L2VPN - and is extensible to support Ethernet segments for other L2VPN types.

- A list of the protocols that are instantiated under the network instance. The protocols that are defined in [I-D.openconfig-rtgwg-local-routing] are included directly such that forwarding entries generated as static or aggregate routes can be matched in a routing policy. Other protocols, such as BGP (for which the data model is defined in [I-D.ietf-idr-bgp-model] would augment this list to include their own configuration options). It should be noted that the list of protocols is not keyed directly on the protocol name, but rather also includes an identifier for that instance. This supports cases where a single network instance may run multiple instances of a protocol.

It is expected that all devices maintain at least one network instance, which represents the default (or global) forwarding table of the device. Additional instances are intended to support virtual instances, such as VSIs and VRFs.

Where base configuration is required for such instances to be utilised (e.g., an route distinguisher value is required) then this is configured globally for each network instance. It expected that additional parameters, such as the encapsulation utilised for the instance will be added in future revisions of the model.

Policies for leaking of routes between instances and tables leverage [I-D.ietf-rtgwg-policy-model] as a policy framework. This allows definition of 'import' policies (allowing a pull model) or export policies (allowing a push model) to be utilised for installation of entries into a local or remote network instance. The same framework is utilised to leak prefixes between multiple tables that are instantiated under an individual network instance.

2.1. Supporting Layer 2 Forwarding Constructs

Whilst the initial focus of parameters in the model is to allow the instantiation of attributes related to Layer 3 forwarding constructs - it is expected that other configuration parameters, such as the definition of 'cross-connections' between local and/or remote interfaces or connection points can be utilised to represent forwarding constructs within the network instance.

The addition of this to the model is still a work-in-progress. The inclusion of the connection point configuration and state parameters is intended to demonstrate how L2 and L3 constructs can co-exist with one another within the model.

3. Open Issues

3.1. Representation of per-protocol tables vs. a single table for all protocols

Some implementations maintain a single table into which all protocols install entries. Each protocol is then configured to explicitly import entries from that table into their local 'export' table. Others maintain multiple RIBs and require explicit connections to be made between these tables such that entries from one protocol are made available to another.

There are essentially two alternatives for how this is modelled in the network-instance model:

1. Allow each protocol to specify a 'target-table' that indicates the table that its entries are to be installed into. In the case that the same target table is specified for multiple protocols, implicit import/export configuration between tables could be created on implementations that implement multiple RIBs.

Explicit configuration can be utilised to explicitly allow an operator to leak between tables in the case that multiple target tables are specified. It should be noted that in fact, protocols that support multiple address families need to allow specification of the target table on a per-address-family basis.

2. Assume that all protocols implement their own target tables. In this case, explicit configuration is utilised to leak between them. Implementations that do not support such per-protocol tables could implicitly generate the configuration that results in forwarding entries being leaked between the tables such that the appearance of a single target-table is maintained.

As per the discussion in Section 2 currently, this model chooses the former implementation approach.

4. Security Considerations

Routing configuration has a significant impact on network operations, and as such any related model carries potential security risks.

YANG data models are generally designed to be used with the NETCONF protocol over an SSH transport. This provides an authenticated and secure channel over which to transfer configuration and operational data. Note that use of alternate transport or data encoding (e.g., JSON over HTTPS) would require similar mechanisms for authenticating and securing access to configuration data.

Most of the data elements in the network intsance model could be considered sensitive from a security standpoint. Unauthorized access or invalid data could cause major disruption.

5. IANA Considerations

This YANG data model and the component modules currently use a temporary ad-hoc namespace. If and when it is placed on redirected for the standards track, an appropriate namespace URI will be registered in the IETF XML Registry" [RFC3688]. The routing policy YANG modules will be registered in the "YANG Module Names" registry [RFC6020].

6. YANG module

The network instance model is described by the YANG module below.

<CODE BEGINS> file openconfig-network-instance.yang

```
module openconfig-network-instance {

  yang-version "1";

  // namespace
  namespace "http://openconfig.net/yang/network-instance";

  prefix "netinst";

  // import some basic types
  //import ietf-inet-types { prefix inet; }
  import ietf-yang-types { prefix "yang"; }
  import ietf-inet-types { prefix "inet"; }
  import openconfig-network-instance-types { prefix "nit"; }
  import openconfig-policy-types { prefix "pt"; }
  import openconfig-routing-policy { prefix "rpol"; }
  import openconfig-local-routing { prefix "lroute"; }
  import openconfig-interfaces { prefix "ocif"; }
  import openconfig-extensions { prefix "ocext"; }

  // meta
  organization "OpenConfig working group";

  contact
    "OpenConfig working group
    www.openconfig.net";

  description
    "An OpenConfig description of a network-instance. This may be
    a Layer 3 forwarding construct such as a virtual routing and
    forwarding (VRF) instance, or a Layer 2 instance such as a
    virtual switch instance (VSI). Mixed Layer 2 and Layer 3
    instances are also supported.";

  ocext:openconfig-version "0.1.0";

  revision "2015-10-18" {
    description
      "Initial revision";
    reference "0.1.0";
  }

  grouping network-instance-top {
    description
      "Top-level grouping containing a list of network instances.";

        container network-instances {
      description
```

```
        "The L2, L3, or L2+L3 forwarding instances that are
        configured on the local system";

                list network-instance {
                        key "name";

        description
          "Network instances configured on the local system";

                        leaf name {
                                type leafref {
                                        path "../config/name";
                                }
                                description
                                        "A unique name identifying the network
instance";
                        }

        container config {
          description
            "Configuration parameters relating to a network
            instance";
          uses network-instance-config;
          uses network-instance-l3vrf-config {
            when "../type = 'L3VRF'" {
              description
                "Layer 3 VRF configuration parameters included when a
                network instance is of type L3VRF";
            }
          }
        }
        container state {
          config false;
          description
            "Operational state parameters relating to a network
            instance";
          uses network-instance-config;
          uses network-instance-l3vrf-config {
            when "../type = 'L3VRF'" {
              description
                "Layer 3 VRF configuration parameters included when a
                network instance is of type L3VRF";
            }
          }
          uses network-instance-state;
        }

        container inter-instance-policies {
```

description

```
            "Policies dictating how RIB or FIB entries are imported
            to and exported from this instance";

          uses rpol:apply-policy-group;
        }

        container table-connections {
          description
            "Policies dictating how RIB or FIB entries are propagated
            between tables";

          list table-connection {
            key "src-table dst-table";

            description
              "A list of connections between pairs of routing or
              forwarding tables, the leaking of entries between
              which is specified by the import and export policy";

            leaf src-table {
              type leafref {
                path "../config/src-table";
              }
              description
                "The name of the table which should be utilised
                as the source of forwarding or routing information";
            }

            leaf dst-table {
              type leafref {
                path "../config/dst-table";
              }
              description
                "The table to which routing entries should be
                exported";
            }

            container config {
              description
                "Configuration parameters relating to the connection
                between tables";
              uses inter-table-policies-config;
            }
            container state {
              config false;
              description
                "State parameters relating to the connection between
                tables";
```

```
            uses inter-table-policies-config;
          }

          uses rpol:apply-policy-group;
        }
      }

      container tables {
        description
          "The routing tables that are managed by this network
          instance";

        list table {
          key "table-name";

          description
            "A network instance manages one or more forwarding or
            routing tables. These may reflect the Layer 2
            forwarding information base, the Layer 3 routing
            information base of the MPLS LFIB. Protocols may be
            explictly associated with a particular table into
            which they populate entries. Multiple protocols may
            install entries into a single table, or there may be a
            1:1 relationship between a routing protocol and a
            table .The import-policy and export-policy lists are
            used to specify how routes leak between different
            tables within the same forwarding instance.";

          leaf table-name {
            type leafref {
              path "../config/table-name";
            }
            description
              "A name for the table";
          }

          container config {
            description
              "Configuration parameters related to the table";
            uses table-config;
          }

          container state {
            config false;
            description
              "State parameters related to the table";
            uses table-config;
          }
```

```
        }
      }

      container interfaces {
        description
          "Interfaces associated with this network intance";

        container config {
          description
            "Configuration parameters relating to interfaces
            associated with the instance";
          uses instance-interfaces-config;
        }
        container state {
          config false;
          description
            "State parameters relating to interfaces associated
            with the instance";
          uses instance-interfaces-config;
          uses instance-interfaces-state;
        }
      }

      container connection-points {
        description
          "The set of connection points within a forwarding
          instance";

        list connection-point {
          key "connection-point-id";

          description
            "A connection point within a Layer 2 network instance.
            Each connection-point consists of a set of interfaces
            only one of which is active at any one time. Other than
            the specification of whether an interface is local
            (i.e., exists within this network-instance), or remote,
            all configuration and state parameters are common";

          leaf connection-point-id {
            type leafref {
              path "../config/connection-point-id";
            }
            description
              "A locally significant reference for the
              connection-point";
          }
```

```
container config {
  description
    "Configuration parameters relating to a Layer 2
    network instance connection point";
  uses instance-connection-point-config;
}
container state {
  config false;
  description
    "Operational state parameters relating to a Layer 2
    network instance connection point";

  uses instance-connection-point-config;
  uses instance-connection-point-state;
}

container endpoints {
  when "../config/type = 'L2P2P' " +
     "or ../config/type = 'L2VSI'" {
    description
      "Configuration parameters to associate interfaces
       into a common group for use in Layer 2 network
       instances";
  }

  description
    "The set of endpoints which are grouped within the
    connection point";

  list endpoint {
    key "endpoint-id";

    description
      "A list of the endpoints (interfaces or remote
      connection points that can be used for this
      connection point). The active endpoint is selected
      based on the precedence that it is configured
      with";

    leaf endpoint-id {
      type leafref {
        path "../config/endpoint-id";
      }
      description
        "A pointer to the configured identifier for the
        endpoint";
    }
```

```
            container config {
              description
                "Configuration parameters relating to the
                endpoint";
              uses instance-endpoint-config;
            }
            container state {
              config false;
              description
                "Operational state parameters relating to the
                endpoint";
              uses instance-endpoint-config;
              uses instance-endpoint-state;
            }
          }
        }
      }
    }

    container protocols {
      description
        "The routing protocols that are enabled for this
        network-instance.";

      list protocol {
        key "identifier name";

        description
          "A process (instance) of a routing protocol. Some
          systems may not support more than one instance of
          a particular routing protocol";

        leaf identifier {
          type leafref {
            path "../config/identifier";
          }
          description
            "The protocol name for the routing or forwarding
            protocol to be instantiated";
        }

        leaf name {
          type leafref {
            path "../config/name";
          }
          description
            "An operator-assigned identifier for the routing
            or forwarding protocol. For some processes this
```

```
            leaf may be system defined.";
          }

          container config {
            description
              "Configuration parameters relating to the routing
              protocol instance";

            uses protocols-config;
          }

          container state {
            config false;
            description
              "State parameters relating to the routing protocol
              instance";

            uses protocols-config;
            uses protocols-state;
          }

          container static {
            when "../config/identifier = 'STATIC'" {
              description
                "Include static route parameters only when the
                protocol is set to static";
            }
            description
              "Configuration and state parameters relating to
              static routes";
            uses lroute:local-static-top;
          }

          container aggregate {
            when "../config/identifier = 'LOCAL-AGGREGATE'" {
              description
                "Include aggregate route parameters only when the
                protocol is set to aggregate";
            }
            description
              "Configuration and state parameters relating to
              locally generated aggregate routes";
            uses lroute:local-aggregate-top;
          }
        }
      }
            }
      }
```

```
  }

  grouping instance-endpoint-config {
    description
      "Configuration data relating to an forwarding-instance
      endpoint";

    leaf endpoint-id {
      type string;
      description
        "An identifier for the endpoint";
    }

    uses instance-endpoint-local-remote;

    leaf precedence {
      type uint16;
      description
        "The precedence of the endpoint - the lowest precendence
        viable endpoint will be utilised as the active endpoint
        within a connection";
    }
  }

  grouping instance-endpoint-local-remote {
    description
      "A generic specification of a local or remote endpoint";

    choice local-remote {
      case local {
        leaf interface {
          type leafref {
            path "/network-instances/network-instance" +
                 "/interfaces/config/interface";
          }
          description
            "Reference to the local interface that is a member of
            the forwarding-instance";
        }
      }
      case remote {
        leaf neighbor {
          type inet:ip-address;
          description
            "The IP address of the device which hosts the
            remote end-point";
        }
```

```
        leaf virtual-circuit-identifier {
          type uint32;
          description
            "The virtual-circuit identifier that identifies the
            connection at the remote end-point";
        }
      }
      description
        "Configuration relating to an endpoint which can either be
        local (an interface), or remote. In the case where it is
        remote a neighbor IP address and virtual-circuit identifier
        must be specified";
    }
  }

  grouping instance-endpoint-state {
    description
      "Operational state data relating to a forwarding-instance
      endpoint";
    leaf active {
      type boolean;
      description
        "When the backup endpoint is active, the value of this
        parameter is set to true";
    }
  }

  grouping instance-connection-point-config {
    description
      "Configuration data relating to a forwarding-instance
      connection point";

    leaf connection-point-id {
      type string;
      description
        "An identifier for a connection point";
    }
  }

  grouping instance-connection-point-state {
    description
      "Operational state data relating to a forwarding-instance
      connection point";
  }

  grouping table-config {
    description
      "Configuration parameters relating to a L2/L2.5/L3 table that
```

```
      exists within the network instance";

    leaf table-name {
      type string;
      description
        "A human-readable name for the table";
    }

  }

  grouping table-state {
    description
      "State parameters relating to a table - this may be
      utilised to store generic structure for retrieving the contents
      of a RIB, FIB or LFIB";
    // placeholder
  }

  grouping inter-table-policies-config {
    description
      "Configuration entries that relate to how RIB or FIB entries
      are propagated between tables within the same network
      instance";

    leaf src-table {
      type leafref {
        // we are at table-connections/table-connection/config/.
        path "../../../../tables/table/table-name";
      }
      description
        "The source protocol for the table connection";
    }

    leaf dst-table {
      type leafref {
        path "../../../../tables/table/table-name";
      }
      description
        "The destination protocol for the table connection";
    }


  }

  grouping network-instance-config {
    description
      "Configuration parameters relating to a top-level network
      instance";
```

```
    leaf name {
      type string;
      description
        "An operator-assigned unique name for the forwarding
        instance";
    }

    leaf type {
      type identityref {
        base "nit:network-instance-type";
      }
      description
        "The type of network instance. The value of this leaf
        indicates the type of forwarding entries that should be
        supported by this network instance";
    }

    leaf enabled {
      type boolean;
      description
        "Whether the network instance should be configured to be
        active on the network element";
    }

    leaf description {
      type string;
      description
        "A free-form string to be used by the network operator to
        describe the function of this network instance";
    }

    leaf router-id {
      type yang:dotted-quad;
      description
        "A identifier for the local network instance - typically
        used within associated routing protocols or signalling
        routing information in another network instance";
    }

    leaf route-distinguisher {
      type nit:route-distinguisher;
      description
        "The route distinguisher that should be used for the local
        VRF or VSI instance when it is signalled via BGP.";
    }
  }

  grouping network-instance-state {
```

```
    description
      "Operational state parameters relating to a network instance";
  }

  grouping network-instance-l3vrf-config {
    description
      "Configuration parameters for a network instance of type
      l3vrf";

  }

  grouping protocols-config {
    description
      "Configuration parameters relating to a generic protocol
      instance within a network instance";

    leaf identifier {
      type identityref {
        base "pt:install-protocol-type";
      }
      description
        "The protocol identifier for the instance";
    }

    leaf name {
      type string;
      description
        "A unique name for the protocol instance";
    }

    leaf enabled {
      type boolean;
      description
        "A boolean value indicating whether the local protocol
        instance is enabled.";
    }

    leaf target-table {
     type leafref {
      path "../../../../tables/table/table-name";
     }
     description
      "The table (RIB, FIB, or LFIB) that the protocol should
      populate its entries in.";
    }
  }

  grouping protocols-state {
```

```
    description
      "Operational state parameters relating to a protocol instance";
  }

  grouping instance-interfaces-config {
    description
      "Base configuration parameters relating to the interfaces
      associated with a network instance";

    leaf-list interface {
      type leafref {
        path "/ocif:interfaces/ocif:interface/ocif:name";
      }
      description
        "Interfaces that are associated with the network instance";
    }
  }

  grouping instance-interfaces-state {
    description
      "State parameters relating to the interfaces associated with a
      network instance";
  }

  uses network-instance-top;

}

<CODE ENDS>


   <CODE BEGINS> file openconfig-network-instance-types.yang

   module openconfig-network-instance-types {

     yang-version "1";

     // namespace
     namespace "http://openconfig.net/yang/network-instance-types";

     prefix "nit";

     import openconfig-extensions { prefix "ocext"; }

     // meta
     organization "OpenConfig working group";

     contact
```

```
    "OpenConfig working group
    www.openconfig.net";

  description
    "Types associated with a network instance";

  ocext:openconfig-version "0.1.0";

  revision "2015-10-18" {
    description
      "Initial revision";
    reference "0.1.0";
  }

  // identity statements
  identity network-instance-type {
        description
         "A base identity which can be extended to indicate different
     types of network instance supported by a device.";
  }

  identity DEFAULT-INSTANCE {
    base network-instance-type;
    description
      "A special routing instance which acts as the 'default' or
      'global' routing instance for a network device.";
  }

  identity L3VRF {
    base network-instance-type;
    description
      "A private Layer 3 only routing instance which is formed of
      one or more RIBs";
  }

  identity L2VSI {
    base network-instance-type;
    description
      "A private Layer 2 only switch instance which is formed of
      one or more L2 forwarding tables";
  }

  identity L2P2P {
    base network-instance-type;
    description
      "A private Layer 2 only forwarding instance which acts as
      a point to point connection between two endpoints";
  }
```

```
    identity L2L3 {
      base network-instance-type;
      description
        "A private Layer 2 and Layer 2 forwarding instance";
    }

    // rjs note:
    // this should move to openconfig-types when merged
    typedef route-distinguisher {
      type union {
        // type 0: <2-byte administrator>:<4-byte assigned number>
        type string {
          pattern "(65[0-5][0-3][0-5]|[1-5][1-5][0-9][0-9][0-9]|"
                  + "[1-9]?[1-9]?[0-9][0-9]|[1-9]):"
                  + "(4[0-2][0-9][0-4][0-9][0-6][0-7][0-2][0-9][0-5]|"
                  + "[0-3][0-9]{9}|[1-9][0-9]{1,8}|[1-9])";
        }
        // type 1: <ip-address>:<2-byte assigned number>
        type string {
          pattern
            "(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}"
            +  "([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5]):"
            +  "(65[0-5][0-3][0-5]|[1-5][1-5][0-9][0-9][0-9]|"
            +  "[1-9]?[1-9]?[0-9][0-9]|[1-9])";
        }
        // type 2: <4-byte as-number>:<2-byte assigned number>
        type string {
          pattern
            "(4[0-2][0-9][0-4][0-9][0-6][0-7][0-2][0-9][0-5]|"
            + "[0-3][0-9]{9}|[1-9][0-9]{1,8}|[1-9]):"
            + "65[0-5][0-3][0-5]|[1-5]{2}[0-9]{3}|"
            + "[1-9]{0,2}[0-9][0-9]|[1-9])";
        }
      }
      description "A route distinguisher value";
      reference "RFC4364";
    }

  }

  <CODE ENDS>
```

7. References

7.1. Normative references

[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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[I-D.openconfig-rtgwg-local-routing] Shaikh, A., George, J., Shakir, R., and F. Chen, "A Data Model for Locally Generated Routes in Service Provider Networks", draft-openconfig-rtgwg-local-routing-00 (work in progress), July 2015.

[I-D.ietf-rtgwg-policy-model] Shaikh, A., rjs@rob.sh, r., D'Souza, K., and C. Chase, "Routing Policy Configuration Model for Service Provider Networks", draft-ietf-rtgwg-policy-model-00 (work in progress), September 2015.

[I-D.ietf-idr-bgp-model] Shaikh, A., Shakir, R., Patel, K., Hares, S., D'Souza, K., Bansal, D., Clemm, A., Alex, A., Jethanandani, M., and X. Liu, "BGP Model for Service Provider Networks", draft-ietf-idr-bgp-model-00 (work in progress), July 2015.

7.2. Informative references

[YANG-REPO] "OpenConfig YANG Data Model Repository - Network Instance", November 2015, <https://github.com/openconfig/public/releases/network-instance>.

Appendix A. Acknowledgements

TODO

Author's Address

Rob Shakir
Jive Communications, Inc.
1275 West 1600 North, Suite 100
Orem, UT 84057

Email: rjs@rob.sh