Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 6, 2016

A. Shaikh
J. George
Google
R. Shakir
BT
F. Chen
Verizon Communications

July 5, 2015

A Data Model for Locally Generated Routes in Service Provider Networks
draft-openconfig-rtgwg-local-routing-00

Abstract

This document defines a YANG data model for configuring and managing locally generated routes in a vendor-neutral way and based on operational best practice. Locally generated routes are those created by configuration, rather than by dynamic routing protocols. Such routes include static routes, locally created aggregate routes for reducing the number of constituent routes that must be advertised, summary routes for IGPs, etc.

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 January 6, 2016.

1. Introduction

This document describes a simple YANG [RFC6020] data model for locally generated routes, i.e., those not created by dynamic routing protocols. These include static routes, locally created aggregate routes, summary routes for IGPs, etc.

1.1. Goals and approach

This model expresses locally generated routes as generically as possible, avoiding configuration of protocol-specific attributes at the time of route creation. This is primarily to avoid assumptions about how underlying router implementations handle route attributes in various routing table data structures they maintain. Hence, the definition of locally generated routes essentially creates 'bare' routes that do not have any protocol- specific attributes.

When protocol-specific attributes must be attached to a route (e.g., communities on a locally defined route meant to be advertised via BGP), the attributes should be attached via a protocol-specific policy after importing the route into the protocol for distribution (again via routing policy). This model is intended to be used with the generic routing policy framework defined in [I-D.shaikh-rtgwg-policy-model].

This model does not aim to be feature complete -- it is a subset of the configuration parameters available in a variety of vendor implementations, but supports widely used constructs for managing local routes. Model development has been primarily driven by examination of actual configurations across operator networks.

While this document provides an overview of the model, the most current and authoritative version is available in the public YANG model repository [YANG-REPO].

2. Model overview

In this initial version of the local routing model, two primary types of locally generated routes are supported, static routes and local aggregates. Static routes are manually configured routes with a defined prefix and next-hop. The model support next-hops specified

as an IP address, interface, or a special defined value, e.g., DISCARD to prevent forwarding for the specified prefix. Aggregate routes are used to summarize constituent routes and avoid having to advertise each constituent individually, and hence increase routing table size. Aggregate routes are not used for forwarding, rather they are "activated" when a constituent route with a valid next hop is present in the IP routing table.

The model structure is shown below:

```
    +--rw local-routes
      +--rw config
      +--ro state
      +--rw static-routes
      |  +--rw static* [prefix]
      |        ...
      +--rw local-aggregates
         +--rw aggregate* [prefix]
               ...
```

Note that the model follows the convention of representing configuration and operational state at the leaves of the tree, based on recommendations in [I-D.openconfig-netmod-opstate]. In this case, the operational state consists primarily of the applied configuration.

3. Interaction with routing policy

In many vendor implementations, local routes may be annotated with various route attributes or policies. For example, when creating a locally generated aggregate route, it is often possible to specify BGP [RFC4271] communities which can then be used when filtering the routes sent to particular neighbors or peer groups. IGP implementations such as IS-IS and OSPF have similar capability to create "summary" routes that serve a similar purpose to BGP aggregates.

Since these and other local routes are conceptually similar from an operator standpoint, there is a desire to create a single model that may be used generically to address a number of use cases. The approach taken in this model is to define locally generated routes as "bare" routes, i.e., without any protocol-specific attributes such as BGP communities, IGP tags, etc. Instead, these attributes are expected to be added via policy when locally generated routes are injected into a particular protocol for subsequent advertisement.

Another important motivation for not including protocol-specific attributes when configuring local routes is that it assumes

implementations can support the association of a variety of attributes with a route. While this may be true in some implementations, others may segregate routing tables for different protocols into different data structures such that it would not be possible to attach attributes from, say BGP, onto an OSPF route. For this reason, we constrain attributes on locally generated routes to be attached via policy as they are imported into different protocols.

For example, consider the case of configuring a new prefix to be advertised to neighbors via BGP. Using the local routing model and the routing policy model in [I-D.shaikh-rtgwg-policy-model], one way to do this is enumerated below:

1. Declare a static route for the advertised prefix using the local routing model (e.g., with a next hop set to DISCARD). This route is placed in the IP routing table.

2. Define a policy to add BGP attributes to the route -- the policy would match on the prefix and the origin of the route (i.e., STATIC) and its action could be to add a BGP community.

3. Apply the BGP policy as an import policy within BGP, e.g., using the apply-policy statement in the policy model, to import the route into BGP.

4. Export the route to neighbors using an export policy as usual, filtering on the added community attribute if appropriate.

The step of creating a policy to add BGP (or other protocol-specific) attributes to the route is optional if an operator simply wishes to export the route to neighbors, as opposed to also filtering on attributes that are assumed to be present on the locally generated route.

This version of the model does support the capability to specify a generic route tag that can be associated with locally generated routes (i.e., both statics and aggregates). The tag can be used to filter routes that are imported into different routing protocols, for example, or to control which routes are exported to a neighbor. The route tagging capability may be refined further as more implementor feedback is incorporated into the model.

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 local routing 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 local routing model is described by the YANG module below.

```
<CODE BEGINS> file local-routing.yang
module local-routing {

  yang-version "1";

  // namespace
  namespace "http://openconfig.net/yang/local-routing";

  prefix "loc-rt";

  // import some basic types
  import ietf-inet-types { prefix inet; }
  import policy-types { prefix pt; }


  // meta
  organization "OpenConfig working group";

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

  description
    "This module describes configuration and operational state data
```

```
      for routes that are locally generated, i.e., not created by
      dynamic routing protocols.  These include static routes, locally
      created aggregate routes for reducing the number of constituent
      routes that must be advertised, summary routes for IGPs, etc.

      This model expresses locally generated routes as generically as
      possible, avoiding configuration of protocol-specific attributes
      at the time of route creation.  This is primarily to avoid
      assumptions about how underlying router implementations handle
      route attributes in various routing table data structures they
      maintain.  Hence, the definition of locally generated routes
      essentially creates 'bare' routes that do not have any protocol-
      specific attributes.

      When protocol-specific attributes must be attached to a route
      (e.g., communities on a locally defined route meant to be
      advertised via BGP), the attributes should be attached via a
      protocol-specific policy after importing the route into the
      protocol for distribution (again via routing policy).";

    revision "2015-05-01" {
      description
        "Initial revision";
      reference "TBD";
    }

    // extension statements

    // feature statements

    // identity statements

    // typedef statements

    typedef local-defined-next-hop {
      type enumeration {
        enum DROP {
          description
            "Discard or black-hole traffic for the corresponding
            destination";
        }
      }
      description
        "Pre-defined next-hop designation for locally generated
        routes";
    }

    // grouping statements
```

```
    grouping local-generic-settings {
      description
        "Generic options that can be set on local routes When
        they are defined";

      leaf set-tag {
        type pt:tag-type;
        description
          "Set a generic tag value on the route. This tag can be
          used for filtering routes that are distributed to other
          routing protocols.";
      }
    }

    grouping local-static-config {
      description
        "Configuration data for static routes.";

      leaf prefix {
        type inet:ip-prefix;
        description
          "Destination prefix for the static route, either IPv4 or
          IPv6.";
      }

      leaf-list next-hop {
        type union {
          type inet:ip-address;
          type local-defined-next-hop;
          type string;
          //TODO: this should be a leafref pointing to a configured
          //interface, but YANG 1.0 does not support leafrefs in a
          //union type.  It should be updated when YANG 1.1 is
          //released.
        }
        description
          "Specify a set of next hops.  Each entry may be an IP
          address, interface, or a single pre-defined next-hop can be
          used, e.g., drop";
      }

      uses local-generic-settings;
    }

    grouping local-static-state {
      description
        "Operational state data for static routes";
    }
```

```
    grouping local-static-top {
      description
        "Top-level grouping for the list of static route definitions";

      container static-routes {
        description
          "Enclosing container for the list of static routes";

        list static {
          key prefix;
          description
            "List of locally configured static routes";

          leaf prefix {
            type leafref {
              path "../config/prefix";
            }
            description
              "Reference to the destination prefix for the static
              route";
          }

          container config {
            description
              "Configuration data for static routes";

            uses local-static-config;
          }

          container state {

            config false;

            description
              "Operational state data for static routes";

            uses local-static-config;
            uses local-static-state;
          }
        }
      }
    }

    grouping local-aggregate-config {
      description
        "Configuration data for aggregate routes";

      leaf prefix {
```

```
        type inet:ip-prefix;
        description
          "Aggregate prefix to be advertised";
      }

      leaf discard {
        type boolean;
        default false;
        description
          "When true, install the aggregate route with a discard
          next-hop -- traffic destined to the aggregate will be
          discarded with no ICMP message generated.  When false,
          traffic destined to an aggregate address when no
          constituent routes are present will generate an ICMP
          unreachable message.";
      }

      uses local-generic-settings;

    }

    grouping local-aggregate-state {
      description
        "Operational state data for local aggregate advertisement
        definitions";
    }

    grouping local-aggregate-top {
      description
        "Top-level grouping for local aggregates";

      container local-aggregates {
        description
          "Enclosing container for locally-defined aggregate
          routes";

        list aggregate {
          key prefix;
          description
            "List of aggregates";

          leaf prefix {
            type leafref {
              path "../config/prefix";
            }
            description
              "Reference to the configured prefix for this aggregate";
          }
```

```
          container config {
            description
              "Configuration data for aggregate advertisements";

            uses local-aggregate-config;
          }

          container state {

            config false;

            description
              "Operational state data for aggregate
              advertisements";

            uses local-aggregate-config;
            uses local-aggregate-state;
          }
        }
      }
    }

    grouping local-routes-config {
      description
        "Configuration data for locally defined routes";
    }

    grouping local-routes-state {
      description
        "Operational state data for locally defined routes";
    }

    grouping local-routes-top {
      description
        "Top-level grouping for local routes";

      container local-routes {
        description
          "Top-level container for local routes";

        container config {
          description
            "Configuration data for locally defined routes";

          uses local-routes-config;
        }

        container state {
```

```
          config false;

          description
            "Operational state data for locally defined routes";

          uses local-routes-config;
          uses local-routes-state;
        }

        uses local-static-top;
        uses local-aggregate-top;
      }
    }

    uses local-routes-top;

  }
  <CODE ENDS>
```

7. Examples

Below we show an example of XML-encoded configuration data using the local routing model, in conjunction with the policy [I-D.shaikh-rtgwg-policy-model] and BGP [I-D.shaikh-idr-bgp-model] models to illustrate how local aggregate routes are defined and distributed into protocols. Although the example focuses on BGP, the aggregate routes may be used with other routing protocols (e.g., IGPs) as mentioned in Section Section 3. Note that the XML has been modified to improve readability.

```
<!-- define an aggregate route -->
<local-routing>
  <local-routes>
    <local-aggregates>
      <aggregate>
        <prefix>10.185.224.0/19</prefix>
        <config>
          <prefix>10.185.224.0/19</prefix>
          <discard>true</discard>
        </config>
      </aggregate>
    </local-aggregates>
  </local-routes>
</local-routing>

<routing-policy>
```

```
    <defined-sets>
      <prefix-sets>
        <prefix-set>
          <prefix-set-name>AGGR_INTERNAL</prefix-set-name>
            <prefix>
              <ip-prefix>10.185.224.0/19</ip-prefix>
              <masklength-range>exact</masklength-range>
            </prefix>
        </prefix-set>
      </prefix-sets>

      <bgp-defined-sets>
        <community-sets>
          <community-set>
            <community-set-name>AGGR_BGP_COMM</community-set-name>
            <community-member>65532:10100</community-member>
            <community-member>65534:60110</community-member>
          </community-set>
        </community-sets>
      </bgp-defined-sets>
    </defined-sets>

    <policy-definitions>

      <!--policy definition to add BGP attributes to the route -->
      <policy-definition>
        <name>ADD_ATTR_BGP_AGGR</name>
        <statements>
          <statement>
            <name>statement-1</name>
            <conditions>
              <install-protocol-eq>LOCAL-AGGREGATE</install-protocol-eq>
              <match-prefix-set>
                  <prefix-set>AGGR_INTERNAL</prefix-set>
              </match-prefix-set>
            </conditions>
            <actions>
              <bgp-actions>
                <set-community>
                  <community-set-ref>AGGR_BGP_COMM</community-set-ref>
                </set-community>
              </bgp-actions>
              <accept-route />
            </actions>
          </statement>
        </statements>
      </policy-definition>
```

```
      <!-- create a policy to export the
      aggregate to UPSTREAM neighbors -->
      <policy-definition>
        <name>EDGE_EXPORT</name>
        <statements>
          <statement>
            <name>statement-2</name>
            <conditions>
              <install-protocol-eq>LOCAL-AGGREGATE</install-protocol-eq>
              <match-prefix-set>
                  <prefix-set>AGGR_INTERNAL</prefix-set>
              </match-prefix-set>
            </conditions>
            <actions>
              <bgp-actions>
                <set-med>100</set-med>
              </bgp-actions>
              <accept-route />
            </actions>
          </statement>
        </statements>
      </policy-definition>
    </policy-definitions>

  </routing-policy>


  <bgp>
    <global>
      <config>
        <as>65518</as>
      </config>
      <!-- apply a policy to import the aggregate into BGP -->
      <apply-policy>
        <config>
          <import-policy>ADD_ATTR_BGP_AGGR</import-policy>
        </config>
      </apply-policy>
    </global>
    <peer-groups>
      <peer-group>
        <peer-group-name>UPSTREAM</peer-group-name>
        <!-- apply a policy to advertise the
        aggregate to the UPSTREAM group -->
        <apply-policy>
          <config>
            <export-policy>EDGE_EXPORT</export-policy>
            <default-export-policy>REJECT_ROUTE</default-export-policy>
```

```
        </config>
      </apply-policy>
    </peer-group>
  </peer-groups>
</bgp>
```

8. References

8.1. Normative references

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2014.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC3688] Mealling, M., "The IETF XML Registry", RFC 3688, January 2004.

8.2. Informative references

[YANG-REPO] "A Data Model for Locally Generated Routes in Service Provider Networks", July 2015, <https://github.com/YangModels/yang/tree/master/experimental/openconfig/local-routing>.

[I-D.shaikh-rtgwg-policy-model] Shaikh, A., Shakir, R., D'Souza, K., and C. Chase, "Routing Policy Configuration Model for Service Provider Networks", draft-shaikh-rtgwg-policy-model-01 (work in progress), July 2015.

[I-D.openconfig-netmod-opstate] Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-00 (work in progress), March 2015.

[I-D.shaikh-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-shaikh-idr-bgp-model-02 (work in progress), June 2015.

Appendix A. Acknowledgements

The authors are grateful for valuable contributions to this document and the associated models from: Phil Bedard, Kevin Brintnall, Matt John, Marcus Hines, and Eric Osborne.

Authors' Addresses

Anees Shaikh
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: aashaikh@google.com

Joshua George
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: jgeorge@google.com

Rob Shakir
BT
pp. C3L, BT Centre
81, Newgate Street
London EC1A 7AJ
UK

Email: rob.shakir@bt.com
URI: http://www.bt.com/

Feihong Chen
Verizon Communications
40 Sylvan Rd.
Waltham, MA
US

Email: feihong.x.chen@verizon.com