A YANG Data Model for Static Route
draft-liang-rtgwg-staticrt-yang-cfg-00

Abstract

This document defines a YANG data model for static routes. The data model includes configuration data and state data.

Requirements Language

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 RFC 2119. [RFC2119]

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 April 18, 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
1. Introduction

Static routes are special routes that are configured manually by network administrators. On simple networks, the network administrator configures static routes in the routers which cannot run dynamic routing protocols or cannot generate routes to a special destination network, so that the network can run properly. Route selection can be controlled using static routes. However, each time a fault occurs on the network or the network topology changes, maybe the network administrator need to reconfigure the static routes.

This document defines a YANG [RFC6020] data model for the configuration and state data of static route. Any RPC or notification definition is not part of this document. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. ReST) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interface, such as CLI and Programmatic APIs.

As many vendors have different object constructs to represent the same static route, it has been tried to design this model in a very flexible, extensible and generic way to fit into most of the vendor requirements.
1.1. Terminology

The following terms are defined in [RFC6020]:

- configuration data
- data model
- module
- state data

The terminology for describing YANG data models is found in [RFC6020].

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration data (read-write), and "ro" means state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Static Route Data Model

The data model has the following structure for configuration of static routes:

```plaintext
module: ietf-staticrt
  +--rw staticrt
    |   +--rw staticrt-cfg
    |     |   +--rw staticrt-topo* [VRF-name address-family topo-name]
    |     |     +--rw topo-name        string
    |     |     +--rw VRF-name        string
    |     |     +--rw address-family  identityref
    |     |     +--rw enable-FRR?     boolean
```
++rw staticrt-entries* [ip-prefix ip-prefix-mask
  output-interface target-VRF-name target-nexthop]
  ++rw ip-prefix     inet:ip-address
  ++rw ip-prefix-mask inet:ip-address
  ++rw output-interface string
  ++rw target-VRF-name string
  ++rw target-nexthop  inet:ip-address
  ++rw description?   string
  ++rw preference?    uint32
  ++rw cost?         uint32
  ++rw tag?         uint32
  ++rw enable-inherit-cost?  boolean
  ++rw is-permanent? boolean
  ++rw not-advertise?  boolean
  ++rw is-iterated-to-host-route?  boolean
  ++rw BFD-enable?    boolean
  ++rw BFD-session-name? string
  ++rw administrated-down-not-selected?  boolean
++ro staticrt-state
++ro staticrt-rib
  ++ro staticrt-route*
    ++ro index?  uint32
    ++ro topo-name? string
    ++ro VRF-name? string
    ++ro address-family? string
    ++ro ip-prefix     inet:ip-address
    ++ro ip-prefix-mask inet:ip-address
    ++ro output-interface? string
    ++ro target-VRF-name? string
    ++ro target-nexthop?  inet:ip-address
    ++ro description?   string
    ++ro preference?    uint32
    ++ro cost?         uint32
    ++ro tag?         uint32
    ++ro enable-inherit-cost?  boolean
    ++ro is-permanent? boolean
    ++ro not-advertise?  boolean
    ++ro is-iterated-to-host-route?  boolean
    ++ro BFD-enable?    boolean
    ++ro BFD-session-name? string
    ++ro administrated-down-not-selected?  boolean
    ++ro iterated-nexthop?  inet:ip-address
    ++ro iterated-output-interface? string
    ++ro label?     uint32
    ++ro route-state? string
    ++ro interface-state? string
    ++ro BFD-state?   string
    ++ro local-address?  inet:ip-address
This data model defines the configuration and state containers for static routes. In staticrt-cfg container, there is a list of configuration containers per static route, which contains the configuration for static route.

The data model for state of static routes defines two state containers. Container staticrt-rib contains the current state of static routes. In the second container, there is statistics information for static routes. The staticrt-entries in the staticrt-statistics container is listed per routing instance per address family. The list of staticrt-entries is in order of applied rules in the forwarding path.
This data model also defines the configuration for BFD session using to detect the configured next-hop reachability of the static route.

3. Static Route YANG Module

<CODE BEGINS> file "ietf-staticrt@2015-10-16.yang"

module ietf-staticrt{
    namespace "urn:ietf:params:xml:ns:yang:ietf-staticrt";
    prefix staticrt;

    import ietf-inet-types {
        prefix inet;
    }

    organization "TBD";
    contact "TBD";

    description
        "This module contains a collection of YANG definitions for configuring static routes.";

    revision 2015-10-17 {
        description
            "Initial revision.";
        reference
            " [draft-ietf-netmod-routing-cfg-16]
              A YANG Data Model for Routing Management."
    }

    identity address-family {
        description
            "Base identity from which identities describing address families are derived.";
    }

    identity ipv4 {
        base address-family;
        description
            "This identity represents IPv4 address family.";
    }

    identity ipv6 {
        base address-family;
        description
            "This identity represents IPv6 address family.";
    }

<CODE ENDS>
grouping staticrt-entry {
  description "This group defines the static route.";

  leaf ip-prefix {
    type inet:ip-address;
    mandatory "true";
    description "The destination ip prefix of the static route.";
  }

  leaf ip-prefix-mask {
    type inet:ip-address;
    mandatory "true";
    description "The destination ip prefix mask of the static route.";
  }

  leaf output-interface {
    type string;
    description "The name of the output interface.";
  }

  leaf target-VRF-name {
    type string;
    description "The VRF name of the target nexthop.";
  }

  leaf target-nexthop {
    type inet:ip-address;
    description "The configured target nexthop.";
  }

  leaf description {
    type string;
    description "The description of the static route.";
  }

  leaf preference {
    type uint32;
    description "Specifies route preference.";
  }
}
leaf cost {
    type uint32;
    description "Specifies route cost.";
}

leaf tag {
    type uint32;
    description "Specifies route tag.";
}

leaf enable-inherit-cost {
    type boolean;
    default false;
    description
        "The flag indicates whether the static route should
         inherit the cost of the iterated route."
}

leaf is-permanent {
    type boolean;
    default false;
    description
        "The flag indicates whether the static route should
         be released permanently."
}

leaf not-advertise {
    type boolean;
    default false;
    description
        "The flag indicates whether the router should not
         advertise the static route."
}

leaf is-iterated-to-host-route {
    type boolean;
    default false;
    description
        "The flag indicates whether the static route should be
         iterated to a host route."
}

grouping staticrt-BFD-paras {
    description
        "This group defines the corresponding BFD session
         information for detecting the configured nexthop
         reachability of the static route.";
}
leaf BFD-enable {
    type boolean;
    description "The flag indicates whether the router enable
    the corresponding BFD session for this static route.";
}

leaf BFD-session-name {
    type string;
    description "The BFD session name.";
}

leaf adminstated-down-not-selected {
    type boolean;
    description "If this flag is true, it indicates that the corresponding
    static route should not be selected when the BFD session
    associated with it is in the AdminDown State.";
}

grouping staticrt-reachability-paras {
    description "This group defines the reachability detecting session
    information for detecting the configured nexthop
    reachability of the static route.";

    uses staticrt-BFD-paras;
}

grouping staticrt-BFD-state {
    description "This group defines the corresponding BFD session state
    as a detecting result of the configured nexthop
    reachability.";

    leaf BFD-state {
        type string;
        config false;
        description "The BFD session state.";
    }

    leaf local-address {
        type inet:ip-address;
        config false;
        description "the local IP address of the corresponding BFD session.";
    }
}
leaf remote-address {
    type inet:ip-address;
    config false;
    description
        "the remote IP address of the corresponding BFD session.";
}

grouping staticrt-reachability-state {
    description
        "This group defines the reachability detecting session
        state as a detecting result of the configured nexthop
        reachability.";

    uses staticrt-BFD-state;
}

container staticrt {
    description
        "Container for static route configuration and state";
    container staticrt-cfg {
        description
            "Configuration for static route.";
    list staticrt-topo {
        key "VRF-name address-family topo-name";
        description
            "Configuration of a static route list.";

        leaf topo-name {
            type string;
            description
                "The topology name of the destination ip prefix
                belonging to.";
        }

        leaf VRF-name {
            type string;
            description
                "The VRF-name of the staticrt-entry";
        }

        leaf address-family {
            type identityref {
                base address-family;
            }
            description
                "address-family of the staticrt-entry";
        }

        leaf enable-FRR {

type boolean;
description
"Enable FRR. FRR is implemented only on static routes that are manually configured. That is, FRR is not implemented on iterated next hops."
}

list staticrt-entries {
    key "ip-prefix ip-prefix-mask output-interface"
        "target-VRF-name target-nexthop";
    ordered-by system;
    uses staticrt-entry;
    uses staticrt-reachability-paras;

description
"Define static routes."
}
}
}

container staticrt-state {
    config false;
    description
"Operational state of static routes."
}

container staticrt-rib{
    description
"Define the operational state data for static routes."
}

list staticrt-route {
    description
"Static routes are organized into list of routes."
}

leaf index {
    type uint32;
    description
"Static route entry index."
}

leaf topo-name {
    type string;
    description
"The topology name of the destination ip prefix belonging to."
}

leaf VRF-name {
    type string;
description
    "VRF-name of the set of static routes";
}

leaf address-family {
    type string;
    description
        "Address-family of the set of static-routes";
}

uses staticrt-entry;
uses staticrt-reachability-paras;

leaf iterated-nexthop {
    type inet:ip-address;
    config false;
    description "The iterated nexthop.";
}

leaf iterated-output-interface {
    type string;
    config false;
    description "The iterated output interface name.";
}

leaf label {
    type uint32;
    config false;
    description "Specifies egress label.";
}

leaf route-state {
    type string;
    config false;
    description "Route state.";
}

leaf interface-state {
    type string;
    config false;
    description "The configured output interface state.";
}

uses staticrt-reachability-state;
}
}

container staticrt-statistics {
    config false;
    description
"Define the statistics of list of static routes."

list staticrt-stats {
  description
  "Statistics of list of static routes per VRF & 
  Address-family & topology.";

  leaf topo-name {
    type string;
    description
    "The topology name of the destination ip prefix 
    belonging to.";
  }

  leaf VRF-name {
    type string;
    description
    "Vrf-name of the set of static routes";
  }

  leaf address-family {
    type string;
    description
    "Address-family of the set of static-routes";
  }

list static-route-stats {
  description
  "This defines the static route statistics of 
  each static route.";

  uses staticrt-entry;

  leaf classified-pkts {
    type uint64;
    description
    " Number of total packets which matched 
    to the static route";
  }

  leaf classified-bytes {
    type uint64;
    description
    " Number of total bytes which matched 
    to the static route";
  }
}
container BFD-paras-cfg {
    description
    "This container defines the corresponding BFD session
    information for detecting the configured nexthop
    reachability of the static route.";

    leaf BFD-enable {
        type boolean;
        description
        "The flag indicates whether the router enable
        the corresponding BFD session for this static route.";
    }

    list BFD-session-paras {
        key "BFD-session-name";
        description "The BFD sessions for static routes";

        leaf BFD-session-name {
            type string;
            description "The BFD session name.";
        }

        leaf address-family {
            type identityref {
                base address-family;
            }
            description
            " address-family of the staticrt-entry";
        }

        leaf interface-name {
            type string;
            description "Interface name.";
        }

        leaf destination-VRF-name {
            type string;
            description "Destination vpn instance name for Gateway.";
        }

        leaf nexthop {
            type inet:ip-address;
            description "NextHop address.";
        }

        leaf local-ip-address {
            type inet:ip-address;
            description "Local IP address for BFD session.";
        }
    }
}
4. IANA Considerations

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in [RFC3688], the following registration has been made.


Registrant Contact: The RTGWG WG of the IETF.

XML: N/A; the requested URI is an XML namespace.

This document registers a YANG module in the "YANG Module Names" registry [RFC6020].

Name: ietf-staticrt


Prefix: staticrt

Reference: RFC xxxx
5. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in the YANG module which are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., <edit-config>) to these data nodes without proper protection can have a negative effect on network operations.

6. Acknowledgements

The editor of this document wishes to thank Eric Wu, Guangying Zheng, Baohua Song, Xiangfeng Ding for the guidance and support in coming up with this draft.

7. References

7.1. NormativeInformative


7.2. Informative References


Authors’ Addresses

Qiandeng Liang
Huawei Technologies
101 Software Avenue, Yuhuatai District
Nanjing  210012
China

Email: liangqiandeng@huawei.com

Gang Yan
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing  100095
China

Email: yangang@huawei.com

Shunwan Zhuang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing  100095
China

Email: zhuangshunwan@huawei.com