RIFT WG
Internet-Draft
Intended status: Standards Track
Expires: November 22, 2019

Zheng. Zhang
Yuehua. Wei
ZTE Corporation
Shaowen. Ma
Mellanox
Xufeng. Liu
Volta Networks

May 21, 2019

RIFT YANG Model
draft-ietf-rift-yang-00

Abstract

This document defines a YANG data model for the configuration and management of RIFT Protocol.

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 https://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 November 22, 2019.

Table of Contents

1. Introduction

[I-D.ietf-rift-rift] introduces the protocol definition of RIFT. This document defines a YANG data model that can be used to configure and manage the RIFT protocol. The model is based on YANG 1.1 as defined in [RFC7950] and conforms to the Network Management Datastore Architecture (NDMA) as described in [RFC8342]

2. Design of the Data Model

This model imports and augments ietf-routing YANG model defined in [RFC8349]. Both configuration branch and state branch of [RFC8349] are augmented. The configuration branch covers node base and policy configuration. The neighbor state will be added in later version. The container "rift" is the top level container in this data model. The presence of this container is expected to enable RIFT protocol functionality.

```
module: ietf-rift
  augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:
    +--rw rift!
       +--rw node-info
       |  +--rw level?                  level
       |  +--rw systemid                systemid
       |  +--rw name?                   string
       |  +--rw pod?                    uint32
       |  +--rw overload?               boolean
       |  +--rw flood-reducing?         boolean {flood-reducing}?
       |  +--rw hierarchy-indications?  enumeration
       |  +--rw interfaces* [local-id]
```

```
       |  |  +--rw local-id                  linkidtype
       |  |  +--rw name?                     if:interface-ref
       |  |  +--rw if-index?                 if:interface-ref
       |  |  +--rw bfd?                      boolean {bfd}?
       |  |  +--rw direction-type?           enumeration
       |  |  +--rw you_are_flood_repeater?   boolean
       |  |  +--rw not_a_ztp_offer?          boolean
       |  |  +--rw flood-port?               inet:port-number
       |  |  +--rw lie-rx-port?              inet:port-number
       |  |  +--rw holdtime?                 rt-types:timer-value-seconds16
       |  |  +--rw local-nonce?              uint16
       |  +--rw (algorighm-type)?
       |  |  +--:(spf)
       |  |  +--:(all-path)
       |  +--ro hal?                      level
       |  +--ro vol-list
       |  |  +--ro vol* [systemid]
       |  |     +--ro offered-level?           level
       |  |     +--ro level?                   level
       |  |     +--ro systemid                 systemid
       |  |     +--ro name?                    string
       |  |     +--ro pod?                     uint32
       |  |     +--ro overload?                boolean
       |  |     +--ro flood-reducing?          boolean {flood-reducing}?
       |  |     +--ro hierarchy-indications?   enumeration
       |  |     +--ro interfaces* [local-id]
       |  |        +--ro local-id                  linkidtype
       |  |        +--ro name?                     if:interface-ref
       |  |        +--ro if-index?                 if:interface-ref
       |  |        +--ro bfd?                      boolean {bfd}?
       |  |        +--ro direction-type?           enumeration
       |  |        +--ro you_are_flood_repeater?   boolean
       |  |        +--ro not_a_ztp_offer?          boolean
       |  |        +--ro flood-port?               inet:port-number
       |  |        +--ro lie-rx-port?              inet:port-number
       |  |        +--ro holdtime?                 rt-types:timer-value-
seconds16
       |  |        +--ro local-nonce?              uint16
       |  +--ro miscabled-links*          linkidtype
       +--ro neighbor
       |  +--ro nbrs* [systemid remote-id]
       |     +--ro level?                   level
       |     +--ro systemid                 systemid
       |     +--ro name?                    string
       |     +--ro pod?                     uint32
       |     +--ro overload?                boolean
       |     +--ro flood-reducing?          boolean {flood-reducing}?
       |     +--ro hierarchy-indications?   enumeration
       |     +--ro interfaces* [local-id]
```

```
       |     |  +--ro local-id                  linkidtype
       |     |  +--ro name?                     if:interface-ref
       |     |  +--ro if-index?                 if:interface-ref
       |     |  +--ro bfd?                      boolean {bfd}?
       |     |  +--ro direction-type?           enumeration
       |     |  +--ro you_are_flood_repeater?   boolean
       |     |  +--ro not_a_ztp_offer?          boolean
       |     |  +--ro flood-port?               inet:port-number
       |     |  +--ro lie-rx-port?              inet:port-number
       |     |  +--ro holdtime?                 rt-types:timer-value-seconds16
       |     |  +--ro local-nonce?              uint16
       |     +--ro remote-id                uint32
       |     +--ro local-id?                uint32
       |     +--ro distance?                uint32
       |     +--ro remote-nonce?            uint16
       |     +--ro miscabled-links*         linkidtype
       +--ro database
       |  +--ro ties* [tie-index]
       |     +--ro tie-index       uint32
       |     +--ro database-tie
       |        +--ro originator?       systemid
       |        +--ro direction-type?   enumeration
       |        +--ro tie-number?       uint32
       |        +--ro seq?              uint32
       |        +--ro lifetime?         uint16
       |        +--ro tie-node
       |        |  +--ro level?                   level
       |        |  +--ro systemid                 systemid
       |        |  +--ro name?                    string
       |        |  +--ro pod?                     uint32
       |        |  +--ro overload?                boolean
       |        |  +--ro flood-reducing?          boolean {flood-reducing}?
       |        |  +--ro hierarchy-indications?   enumeration
       |        |  +--ro interfaces* [local-id]
       |        |     +--ro local-id                  linkidtype
       |        |     +--ro name?                     if:interface-ref
       |        |     +--ro if-index?                 if:interface-ref
       |        |     +--ro bfd?                      boolean {bfd}?
       |        |     +--ro direction-type?           enumeration
       |        |     +--ro you_are_flood_repeater?   boolean
       |        |     +--ro not_a_ztp_offer?          boolean
       |        |     +--ro flood-port?               inet:port-number
       |        |     +--ro lie-rx-port?              inet:port-number
       |        |     +--ro holdtime?                 rt-types:timer-value-
seconds16
       |        |     +--ro local-nonce?              uint16
       |        +--ro tie-prefix
       |        |  +--ro prefixes
       |        |  |  +--ro prefix?            inet:ip-prefix
```

```
       |        |  |  +--ro metric?            uint32
       |        |  |  +--ro tag?               uint64
       |        |  |  +--ro monotonic_clock?   PrefixSequenceType
       |        |  |  +--ro from-link?         linkidtype
       |        |  +--ro positive_disaggregation_prefixes
       |        |  |  +--ro prefix?            inet:ip-prefix
       |        |  |  +--ro metric?            uint32
       |        |  |  +--ro tag?               uint64
       |        |  |  +--ro monotonic_clock?   PrefixSequenceType
       |        |  |  +--ro from-link?         linkidtype
       |        |  +--ro negative_disaggregation_prefixes
       |        |  |  +--ro prefix?            inet:ip-prefix
       |        |  |  +--ro metric?            uint32
       |        |  |  +--ro tag?               uint64
       |        |  |  +--ro monotonic_clock?   PrefixSequenceType
       |        |  |  +--ro from-link?         linkidtype
       |        |  +--ro external_prefixes
       |        |     +--ro prefix?            inet:ip-prefix
       |        |     +--ro metric?            uint32
       |        |     +--ro tag?               uint64
       |        |     +--ro monotonic_clock?   PrefixSequenceType
       |        |     +--ro from-link?         linkidtype
       |        +--ro kvs
       |           +--ro key?     uint16
       |           +--ro value?   uint32
       +--rw kv-store
          +--rw kvs* [kvs-index]
             +--rw kvs-index    uint32
             +--rw kvs-tie
                +--rw originator?       systemid
                +--rw direction-type?   enumeration
                +--rw tie-number?       uint32
                +--rw key?              uint16
                +--rw value?            uint32

  notifications:
    +---n error-set
       +--ro tie-level-error
       |  +--ro originator?       systemid
       |  +--ro direction-type?   enumeration
       |  +--ro tie-number?       uint32
       |  +--ro seq?              uint32
       |  +--ro lifetime?         uint16
       |  +--ro tie-node
       |  |  +--ro level?                  level
       |  |  +--ro systemid                systemid
       |  |  +--ro name?                   string
       |  |  +--ro pod?                    uint32
```

```
|  |  +--ro overload?                boolean
|  |  +--ro flood-reducing?          boolean {flood-reducing}?
|  |  +--ro hierarchy-indications?   enumeration
|  |  +--ro interfaces* [local-id]
|  |     +--ro local-id                  linkidtype
|  |     +--ro name?                     if:interface-ref
|  |     +--ro if-index?                 if:interface-ref
|  |     +--ro bfd?                      boolean {bfd}?
|  |     +--ro direction-type?           enumeration
|  |     +--ro you_are_flood_repeater?   boolean
|  |     +--ro not_a_ztp_offer?          boolean
|  |     +--ro flood-port?               inet:port-number
|  |     +--ro lie-rx-port?              inet:port-number
|  |     +--ro holdtime?                 rt-types:timer-value-seconds16
|  |     +--ro local-nonce?              uint16
|  +--ro tie-prefix
|  |  +--ro prefixes
|  |  |  +--ro prefix?            inet:ip-prefix
|  |  |  +--ro metric?            uint32
|  |  |  +--ro tag?               uint64
|  |  |  +--ro monotonic_clock?   PrefixSequenceType
|  |  |  +--ro from-link?         linkidtype
|  |  +--ro positive_disaggregation_prefixes
|  |  |  +--ro prefix?            inet:ip-prefix
|  |  |  +--ro metric?            uint32
|  |  |  +--ro tag?               uint64
|  |  |  +--ro monotonic_clock?   PrefixSequenceType
|  |  |  +--ro from-link?         linkidtype
|  |  +--ro negative_disaggregation_prefixes
|  |  |  +--ro prefix?            inet:ip-prefix
|  |  |  +--ro metric?            uint32
|  |  |  +--ro tag?               uint64
|  |  |  +--ro monotonic_clock?   PrefixSequenceType
|  |  |  +--ro from-link?         linkidtype
|  |  +--ro external_prefixes
|  |     +--ro prefix?            inet:ip-prefix
|  |     +--ro metric?            uint32
|  |     +--ro tag?               uint64
|  |     +--ro monotonic_clock?   PrefixSequenceType
|  |     +--ro from-link?         linkidtype
|  +--ro kvs
|     +--ro key?     uint16
|     +--ro value?   uint32
+--ro nbr-error
   +--ro nbrs* [systemid remote-id]
      +--ro level?                  level
      +--ro systemid                systemid
      +--ro name?                   string
```

```
            +--ro pod?                     uint32
            +--ro overload?                boolean
            +--ro flood-reducing?          boolean {flood-reducing}?
            +--ro hierarchy-indications?   enumeration
            +--ro interfaces* [local-id]
            |  +--ro local-id                  linkidtype
            |  +--ro name?                     if:interface-ref
            |  +--ro if-index?                 if:interface-ref
            |  +--ro bfd?                      boolean {bfd}?
            |  +--ro direction-type?           enumeration
            |  +--ro you_are_flood_repeater?   boolean
            |  +--ro not_a_ztp_offer?          boolean
            |  +--ro flood-port?               inet:port-number
            |  +--ro lie-rx-port?              inet:port-number
            |  +--ro holdtime?                 rt-types:timer-value-seconds16
            |  +--ro local-nonce?              uint16
            +--ro remote-id                uint32
            +--ro local-id?                uint32
            +--ro distance?                uint32
            +--ro remote-nonce?            uint16
            +--ro miscabled-links*         linkidtype
```

3. RIFT configuration

RIFT configurations require node base information configurations. Some features can be used to enhance protocol, such as BFD, flooding-reducing, community attribute.

4. RIFT State

RIFT states are composed of RIFT node state, neighbor state, database.

5. RPC

TBD.

6. Notifications

Unexpected TIE and neighbor's layer error should be notified.

7. RIFT YANG model

<CODE BEGINS> file "ietf-rift.yang"

```
module ietf-rift {

 yang-version 1.1;
```

```
    namespace "urn:ietf:params:xml:ns:yang:ietf-rift";
    prefix rift;

    import ietf-inet-types {
        prefix "inet";
        reference "RFC6991";
    }

    import ietf-routing {
        prefix "rt";
        reference "RFC8349";
    }

    import ietf-interfaces {
        prefix "if";
        reference "RFC7223";
    }

    import ietf-routing-types {
        prefix "rt-types";
        reference "RFC8294";
    }

    organization
        "IETF RIFT(Routing In Fat Trees) Working Group";

    contact
        "WG Web:   <http://tools.ietf.org/wg/rift/>
         WG List:  <mailto:rift@ietf.org>

         Editor:    Zheng Zhang
                   <mailto:zzhang_ietf@hotmail.com>

         Editor:    Yuehua Wei
                   <mailto:wei.yuehua@zte.com.cn>

         Editor:    Shaowen Ma
                   <mailto:mashaowen@gmail.com>

         Editor:    Xufeng Liu
                   <mailto:Xufeng_Liu@jabil.com>";

         description
             "The module defines the YANG definitions for RIFT.

          Copyright (c) 2018 IETF Trust and the persons
          identified as authors of the code.  All rights reserved.
```

```
          Redistribution and use in source and binary forms, with or
          without modification, is permitted pursuant to, and subject
          to the license terms contained in, the Simplified BSD License
          set forth in Section 4.c of the IETF Trust's Legal Provisions
          Relating to IETF Documents
          (http://trustee.ietf.org/license-info).
          This version of this YANG module is part of
          draft-ietf-rift-rift; see the draft itself for full legal
          notices.";

    revision 2019-05-05 {
        description "Initial revision.";
        reference
            "RFC XXXX: A YANG Data Model for RIFT.
             draft-ietf-rift-rift: RIFT: Routing in Fat Trees.";
    }

    /*
     * Features
     */

    /*feature overload {
        description "A node with overload bit set SHOULD NOT advertise any
reachability
                     prefixes southbound except locally hosted ones. The leaf
node SHOULD
                     set the 'overload' bit on its node TIEs.";
    }*/

    feature bfd {
        description "Support BFD (RFC5881) function to react quickly to link
failures.";
    }

    feature flood-reducing {
        description "Support flood reducing function defined in section
4.2.3.8.";
    }

    feature policy {
        description "Support policy guide information.";
    }

    typedef systemid {
        type string {
            pattern
            '[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}';
        }
        description
           "This type defines RIFT system id using pattern,
            system id looks like : 0143.0438.0100.AeF0";
```

```
    }
```

```
    typedef level {
        type uint8 {
            range "0 .. 24";
        }
        default "0";
        description "The value of node level. The max value is 24.";
    }

    typedef linkidtype {
       type uint32;
       description
         "This type defines the link id of an interface.";
    }

    typedef PrefixSequenceType {
       type uint64;
       description
         "This type defines the link id of an interface.";
    }

    /*
     * Identity
     */
    identity rift {
        base rt:routing-protocol;
        description "Identity for the RIFT routing protocol.";
    }

    /*
     * Groupings
     */
    grouping node-key {
        leaf systemid {
            type systemid;
            mandatory true;
            description "Each node is identified via a SystemID which is 64
bits wide.";
        }
        description "The key information used to distinguish a node.";
    }

    grouping base-node-info {
        leaf level {
            type level;
            description "The level of this node.";
        }
        uses node-key;

        leaf name {
```

```
            type string;
            description "The name of this node. It won't be used as the key of
node,
                         just used for description.";
        }
        leaf pod {
            type uint32;
            description "Point of Delivery. The self-contained vertical slice of
                         a Clos or Fat Tree network containing normally only
                         level 0 and level 1 nodes. It communicates with nodes
                         in other PoDs via the spine. We number PoDs to
distinguish
                         them and use PoD #0 to denote 'undefined' PoD.";
        }
        leaf overload {
            type boolean;
            description "If the overload bit in TIEs should be set.";
        }
        leaf flood-reducing {
            if-feature flood-reducing;
            type boolean;
            description "If the node support flood reducing function defined in
section 4.2.3.8.";
        }
        uses hierarchy-indications;
        uses interface;

        description "The base information of a node.";
    }

    grouping neighbor {
        leaf remote-id {
            type uint32;
            description "The remote-id to reach this neighbor.";
        }
        leaf local-id {
            type uint32;
            description "The local-id of link connect to this neighbor.";
        }
        leaf distance {
            type uint32;
            description "The cost value to arrive this neighbor.";
        }
        leaf remote-nonce {
            type uint16;
            description "Remote Nonce of the adjacency as received
                         in LIEs.  In case of LIE packet this MUST
                         correspond to the value in the serialized
                         object otherwise the packet MUST be discarded.";
        }
        leaf-list miscabled-links {
```

```
            type linkidtype;
            description "List of miscabled links.";
        }
        description "The neighbor information.";
    }

    grouping hierarchy-indications {
        leaf hierarchy-indications {
            type enumeration {
                enum "leaf-only" {
                    description "The node will never leave the 'bottom of the
                                 hierarchy'.";
                }
                enum "leaf_only_and_leaf_2_leaf_procedures" {
                    description "This means leaf to leaf.";
                }
                enum "top_of_fabric" {
                    description "The node is 'top of fabric'.";
                }
            }
            description "The hierarchy indications of this node.";
        }
        description "The hierarchy indications of this node.";
    }

    grouping node {
        uses base-node-info;
        uses algorithm;

        leaf hal {
            type level;
            config false;
            description "The highest defined level value seen from all
                         valid level offers received.";
        }
        container vol-list {
            config false;
            list vol {
                key "systemid";
                leaf offered-level {
                    type level;
                    description "The level type value offered by this
neighbor.";
                }
                uses base-node-info;
                description "The valid offered level information.";
            }
            description "The valid offered level information.";
        }
```

```
        leaf-list miscabled-links {
            type linkidtype;
            config false;
            description
                "List of miscabled links.";
        }
        description "The information of local node. Includes base information,
                     configurable parameters and features.";
    }

    grouping direction-type {
        leaf direction-type {
            type enumeration {
                enum illegal {
                    description "Illegal direction.";
                }
                enum south {
                    description "A link to a node one level down.";
                }
                enum north {
                    description "A link to a node one level up.";
                }
                enum east-west {
                    description "A link to a node in the same level.";
                }
                enum max {
                    description "The max value of direction.";
                }
            }
            description "The type of a link.";
        }
        description "The type of a link.";
    }

    grouping interface {
        list interfaces {
            key "local-id";
            leaf local-id {
                type linkidtype;
                mandatory true;
                description "The local id of this interface.";
            }
            leaf name {
                type if:interface-ref;
                description "The interface's name.";
            }
            leaf if-index {
                type if:interface-ref;
```

```
                description "The index of this interface.";
            }
            leaf bfd {
                if-feature bfd;
                type boolean;
                description "If BFD function is enabled to react link failures
                             after neighbor's detection.";
            }
            uses direction-type;

            leaf you_are_flood_repeater {
                type boolean;
                description "If the neighbor on this link is flooding
repeater.";
            }
            leaf not_a_ztp_offer {
                type boolean;
                description "If the neighbor on this link offers ZTP.";
            }
            leaf flood-port {
                type inet:port-number;
                description "The flooding port.";
            }
            leaf lie-rx-port {
                type inet:port-number;
                description "The port of LIE packet receiving.";
            }
            leaf holdtime {
                type rt-types:timer-value-seconds16;
                units seconds;
                description "The holding time of this adjacency.";
            }
            leaf local-nonce {
                type uint16;
                description "Local Nonce of the adjacency as advertised in LIEs.
                             In case of LIE packet this MUST correspond to the
                             value in the serialized object otherwise the packet
                             MUST be discarded.";
            }

            description "The interface information on this node.";
        }
        description "The interface information.";
    }

    grouping prefix-info {
        leaf prefix {
            type inet:ip-prefix;
            description "The prefix information.";
```

```
        }
        leaf metric {
            type uint32;
            description "The metric of this prefix.";
        }
        leaf tag {
            type uint64;
            description "The tag of this prefix.";
        }
        leaf monotonic_clock {
            type PrefixSequenceType;
            description "The monotonic clock for mobile addresses.";
        }
        leaf from-link {
            type linkidtype;
            description "In case of locally originated prefixes, i.e.
                         interface addresses this can describe which
                         link the address belongs to.";
        }

        description "The detail information of prefix.";
    }

    grouping tie-id {
        leaf originator {
            type systemid;
            description "The originator's systemid of this TIE.";
        }
        uses direction-type;
        leaf tie-number {
            type uint32;
            description "The number of this TIE";
        }
        description "TIE is the acronym for 'Topology Information Element'.
                     TIEs are exchanged between RIFT nodes to describe parts
                     of a network such as links and address prefixes. This is
                     the TIE identification information.";
    }

    grouping key-value {
        leaf key {
            type uint16;
            description "The type of key value combination.";
        }
        leaf value {
            type uint32;
            description "The value of key value combination.";
        }
```

```
        description "The key-value store information.";
    }

    grouping tie-info {
        leaf seq {
            type uint32;
            description "The sequence number of a TIE.";
        }
        leaf lifetime {
            type uint16 {
                range "1 .. 65535";
            }
            description "The lifetime of a TIE.";
        }

        container tie-node {
            uses base-node-info;
            description "The node element information in this TIE.";
        }
        container tie-prefix {
            container prefixes {
                uses prefix-info;
                description "It is the prefixes TIE element.";
            }
            container positive_disaggregation_prefixes {
                uses prefix-info;
                description "It is the positive disaggregation prefixes TIE
element.";
            }
            container negative_disaggregation_prefixes {
                uses prefix-info;
                description "It is the negative disaggregation prefixes
element.";
            }
            container external_prefixes {
                uses prefix-info;
                description "It is the external prefixes element.";
            }
            description "The prefix information in this TIE.";
        }
        container kvs {
            uses key-value;
            description "The key/values in the database.";
        }

        description "TIE is the acronym for 'Topology Information Element'.
                     TIEs are exchanged between RIFT nodes to describe parts
                     of a network such as links and address prefixes. This TIE
                     info is used to indicate the state of this TIE. When the
                     type of this TIE is set to 'node', the node-element is
```

```
                    making sense. When the type of this TIE is set to other
                    types except for 'node', the prefix-info is making sense.";
    }

    grouping algorithm {
        choice algorighm-type {
            case spf {
                description "The algorithm is SPF.";
            }
            case all-path {
                description "The algorithm is all-path.";
            }
            description "The possible algorithm types.";
        }
        description "The computation algorithm types.";
    }

     /*
      * Data nodes
      */
    augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol" {
        when "derived-from-or-self(rt:type, 'rift:rift')" {
            description "This augment is only valid for a routing protocol
instance of RIFT.";
        }
        description "RIFT ( Routing in Fat Trees ) YANG model.";
        container rift {
            presence "Container for RIFT protocol.";
            description "RIFT configuration data.";

            container node-info {
                description "The node information about RIFT.";
                uses node;
            }
            container neighbor {
                config false;
                list nbrs {
                    key "systemid remote-id";
                    uses base-node-info;
                    uses neighbor;
                    description "The information of a neighbor.";
                }
                description "The neighbor's information.";
            }//neighbor

            container database {
                config false;
                list ties {
                    key "tie-index";
```

```
                    leaf tie-index {
                        type uint32;
                        description "The index of a TIE.";
                    }
                    container database-tie {
                        uses tie-id;
                        uses tie-info;

                        description "The TIEs in the database.";
                    }
                    description "The detail information of a TIE.";
                }
                description "The TIEs information in database.";
            }//database

            container kv-store {
                list kvs {
                    key "kvs-index";
                    leaf kvs-index {
                        type uint32;
                        description "The index of a kv pair.";
                    }

                    container kvs-tie {
                        uses tie-id;
                        uses key-value;
                        description "The TIEs in the kv-store.";
                    }
                    description "The information used to distinguish a Key/
Value pair.
                                 When the type of kv is set to 'node', node-
element is
                                 making sense. When the type of kv is set to
other values
                                 except 'node', prefix-info is making sense.";
                }
                description "The Key/Value store information.";
            }//kv-store

        }//rift
    }//augment


    /*
     * RPCs
     */


    /*
     * Notifications
     */
```

```
    notification error-set {
        description "The errors notification of RIFT.";
        container tie-level-error {
            uses tie-id;
            uses tie-info;
            description "The level is undefined in the LIEs.";
        }
        container nbr-error {
            list nbrs {
                key "systemid remote-id";
                uses base-node-info;
                uses neighbor;
                description "The information of a neighbor.";
            }
            description "The neighbor errors set.";
        }
    }
}
<CODE ENDS>
```

8. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that 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.

The RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations.

9. IANA Considerations

The IANA is requested to assign two new URIs from the IETF XML registry ([RFC3688]). Authors are suggesting the following URI:

URI: urn:ietf:params:xml:ns:yang:ietf-rift

Registrant Contact: RIFT WG

XML: N/A, the requested URI is an XML namespace

This document also requests one new YANG module name in the YANG Module Names registry ([RFC6020]) with the following suggestion:

name: ietf-rift

namespace: urn:ietf:params:xml:ns:yang:ietf-rift

prefix: rift

reference: RFC XXXX

10. Contributors

The authors would like to thank Tony Przygienda, Benchong Xu (xu.benchong@zte.com.cn), for their review and valuable contributions.

11. Normative References

[I-D.ietf-rift-rift]
Team, T., "RIFT: Routing in Fat Trees", draft-ietf-rift-rift-05 (work in progress), April 2019.

[I-D.ietf-rtgwg-policy-model]
Qu, Y., Tantsura, J., Lindem, A., and X. Liu, "A YANG Data Model for Routing Policy Management", draft-ietf-rtgwg-policy-model-06 (work in progress), March 2019.

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, DOI 10.17487/RFC6087, January 2011, <https://www.rfc-editor.org/info/rfc6087>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <https://www.rfc-editor.org/info/rfc7223>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <https://www.rfc-editor.org/info/rfc7277>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/info/rfc8177>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

Authors' Addresses

Zheng Zhang
ZTE Corporation

Email: zzhang_ietf@hotmail.com

Yuehua Wei
ZTE Corporation

Email: wei.yuehua@zte.com.cn

Shaowen Ma
Mellanox

Email: mashaowen@gmail.com

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com