Stream:	Internet Engineering Task Force (IETF)				
RFC:	9719				
Category:	Standards Track				
Published:	April 2025				
ISSN:	2070-1721				
Authors:	Z. Zhang	Y. Wei	S. Ma	X. Liu	B. Rijsman
	ZTE Corporation	*ZTE Corporation*	*Google*	*Individual*	*Individual*

RFC 9719
YANG Data Model for Routing in Fat Trees (RIFT)

Abstract

This document defines a YANG data model for the configuration and management of the Routing in Fat Trees (RIFT) Protocol. The model is based on YANG 1.1, which is defined in RFC 7950 and conforms to the Network Management Datastore Architecture (NMDA) as described in RFC 8342.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9719.

Table of Contents

1. Introduction

[RFC9692] introduces the protocol definition of RIFT. This document defines one NMDA-compatible [RFC8342] YANG 1.1 [RFC7950] data model for the management of the RIFT protocol. This model imports and augments the ietf-routing YANG data model defined in [RFC8349].

1.1. Terminology

The following terminology and abbreviations are used in this document and the defined model.

The content is copied from [RFC9692] for reading convenience.

Clos / Fat Tree:
: This document uses the terms "Clos" and "Fat Tree" interchangeably where it always refers to a folded spine-and-leaf topology with possibly multiple Points of Delivery (PoDs) and one or multiple Top of Fabric (ToF) planes.

RIFT:
: Routing in Fat Trees [RFC9692].

LIE:
: This is an acronym for a "Link Information Element" exchanged on all the system's links running RIFT to form *ThreeWay* adjacencies and carry information used to perform RIFT Zero Touch Provisioning (ZTP) of levels.

Point of Delivery (PoD):
: A self-contained vertical slice or subset of a Clos or Fat Tree network normally containing only level 0 and level 1 nodes. A node in a PoD communicates with nodes in other PoDs via the ToF nodes. PoDs are numbered to distinguish them, and PoD value 0 is used to denote "undefined" or "any" PoD.

ThreeWay Adjacency:
: RIFT tries to form a unique adjacency between two nodes over a point-to-point interface and exchange local configuration and necessary RIFT ZTP information. An adjacency is only advertised in Node TIEs and used for computations after it achieved *ThreeWay* state, i.e., both routers reflected each other in LIEs, including relevant security information. Nevertheless, LIEs before *ThreeWay* state is reached may carry RIFT ZTP related information already.

TIEs:
: This is an acronym for a "Topology Information Element". TIEs are exchanged between RIFT nodes to describe parts of a network such as links and address prefixes. A TIE has always a direction and a type. North TIEs (sometimes abbreviated as N-TIEs) are used when dealing with TIEs in the northbound representation, and South TIEs (sometimes abbreviated as S-TIEs) for the southbound equivalent. TIEs have different types, such as node and prefix TIEs.

Top of Fabric (ToF):
: The set of nodes that provide inter-PoD communication and have no northbound adjacencies, i.e., are at the "very top" of the fabric. ToF nodes do not belong to any PoD and are assigned the default PoD value to indicate the equivalent of "any" PoD.

1.2. Conventions Used in This Document

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

1.4. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module as shown in Table 1.

Prefix	YANG Module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
rt	ietf-routing	[RFC8349]
if	ietf-interfaces	[RFC8343]
rt-types	ietf-routing-types	[RFC8294]
iana-rt-types	iana-routing-types	[RFC8294]
key-chain	ietf-key-chain	[RFC8177]

Table 1

2. Design of the Data Model

2.1. Scope of Model

This model can be used to configure and manage the RIFT protocol. The operational state data and statistics can be retrieved by this model. The subscription and push mechanism defined in [RFC8639] and [RFC8641] can be implemented by the user to subscribe to notifications on the data nodes in this model.

The model contains all the basic configuration parameters to operate the protocol. Depending on the implementation choices, some systems may not allow some of the advanced parameters to be configurable. The occasionally implemented parameters are modeled as optional features in this model. This model can be extended, and it has been structured in a way that such extensions can be conveniently made.

The RIFT YANG module augments the /routing/control-plane-protocols/ control-plane-protocol path defined in the ietf-routing module. This model augments the routing module to add RIFT as a control-plane protocol. It then offers the ability to create a list of instances, which it does by declaring 'list rift'. Multiple instances of the protocol are supported by the module by giving each instance a unique name.

2.2. Specification

This model imports and augments ietf-routing YANG model defined in [RFC8349]. The container "rift" is the top-level container in this data model. The container is expected to enable RIFT protocol functionality.

The YANG data model defined in this document conforms to the Network Management Datastore Architecture (NMDA) [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [RFC8407].

2.3. Overview

The RIFT YANG module defined in this document has all the common building blocks for the RIFT protocol.

At a high level, the RIFT YANG model is organized into five elements:

base protocol configuration -- Configuration affecting RIFT protocol-related operations.

interface configuration -- Configuration affecting the interface operations.

neighbor status -- Information of neighbors.

database -- Information of TIEs.

statistics -- Statistics of SPF, interface, and neighbor.

```
module: ietf-rift
  augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol:
    +--rw rift* [name]
       +--rw name          string
       +--rw global
       |  +--ro node-level?                  level
       |  +--rw system-id                    system-id
       |  +--rw fabric-id?                   uint16
       |  +--rw pod?                         uint32
       |  +--rw fabric-prefix?               inet:ip-prefix
       |  +--rw fabric-prefix-advertise?     boolean
       |  +--rw configured-level?            level
       |  +--rw overload
       |  |  +--rw overload?                   boolean
       |  |  +--rw (timeout-type)?
       |  |     +--:(on-startup)
       |  |     |  +--rw on-startup-timeout?
       |  |     |          rt-types:timer-value-seconds16
```

```
        |  |     +--:(immediate)
        |  |        +--rw immediate-timeout?
        |  |                 rt-types:timer-value-seconds16
        |  +--ro proto-major-ver              uint8
        |  +--ro proto-minor-ver              uint16
        |  +--rw node-capabilities
        |  |  +--rw hierarchy-indications?   enumeration
        |  |  +--rw flood-reduction?         boolean
        |  +--rw maximum-nonce-delta?         uint8
        |  |       {nonce-delta-adjust}?
        |  +--rw nonce-increasing-interval?   uint16
        |  +--rw adjusted-lifetime?
        |  |       rt-types:timer-value-seconds16
        |  +--rw rx-lie-multicast-addr
        |  |  +--rw ipv4?   inet:ipv4-address
        |  |  +--rw ipv6?   inet:ipv6-address
        |  +--rw tx-lie-multicast-addr
        |  |  +--rw ipv4?   inet:ipv4-address
        |  |  +--rw ipv6?   inet:ipv6-address
        |  +--rw lie-tx-port?                 inet:port-number
        |  +--rw global-link-capabilities
        |  |  +--rw bfd-capable?             boolean
        |  |  +--rw v4-forwarding-capable?   boolean
        |  |  +--rw mtu-size?                uint32
        |  +--rw tide-generation-interval?
        |  |       rt-types:timer-value-seconds16
        |  +--rw tie-security* [security-type] {tie-security}?
        |  |  +--rw security-type             enumeration
        |  |  +--rw shared?                   boolean
        |  |  +--rw (auth-key-chain)?
        |  |     +--:(auth-key-chain)
        |  |     |  +--rw key-chain?          key-chain:key-chain-ref
        |  |     +--:(auth-key-explicit)
        |  |        +--rw key?                string
        |  |        +--rw crypto-algorithm?   identityref
        |  +--rw inner-security-key-id?       uint8
        |  +--rw algorithm-type?              enumeration
        |  +--ro hal
        |  |  +--ro hal-value?    level
        |  |  +--ro system-ids*   system-id
        |  +--ro miscabled-links*             uint32
        |  +--rw hop-limit?                   uint8
        |  +--rw maximum-clock-delta?         ieee802-1as-timestamp
        +--rw interfaces* [name]
        |  +--ro link-id?                     uint32
        |  +--rw name                         if:interface-ref
        |  +--rw cost?                        uint32
        |  +--rw rx-flood-port?               inet:port-number
        |  +--rw holdtime?
        |  |       rt-types:timer-value-seconds16
        |  +--rw address-families*
        |  |       iana-rt-types:address-family
        |  +--rw advertised-source-addr
        |  |  +--rw ipv4?   inet:ipv4-address-no-zone
        |  |  +--rw ipv6?   inet:ipv6-address-no-zone
        |  +--ro link-direction-type?         enumeration
        |  +--rw broadcast-capable?           boolean
        |  +--rw allow-horizontal-link?       boolean
```

```
        |  +--rw security {link-security}?
        |  |  +--rw security-type?            enumeration
        |  |  +--rw shared?                   boolean
        |  |  +--rw (auth-key-chain)?
        |  |     +--:(auth-key-chain)
        |  |     |  +--rw key-chain?          key-chain:key-chain-ref
        |  |     +--:(auth-key-explicit)
        |  |        +--rw key?                string
        |  |        +--rw crypto-algorithm?   identityref
        |  +--rw security-checking?           enumeration
        |  +--ro was-the-last-lie-accepted?   boolean
        |  +--ro last-lie-reject-reason?      string
        |  +--ro advertised-in-lies
        |  |  +--ro label?                         uint32
        |  |  |       {label-switching}?
        |  |  +--ro you-are-flood-repeater?        boolean
        |  |  +--ro not-a-ztp-offer?               boolean
        |  |  +--ro you-are-sending-too-quickly?   boolean
        |  +--rw link-capabilities
        |  |  +--rw bfd-capable?             boolean
        |  |  +--rw v4-forwarding-capable?   boolean
        |  |  +--rw mtu-size?                uint32
        |  +--ro state                         enumeration
        |  +--ro neighbors* [system-id]
        |  |  +--ro node-level?                   level
        |  |  +--ro system-id                     system-id
        |  |  +--ro fabric-id?                    uint16
        |  |  +--ro pod?                          uint32
        |  |  +--ro proto-major-ver?              uint8
        |  |  +--ro proto-minor-ver?              uint16
        |  |  +--ro sent-offer
        |  |  |  +--ro level?             level
        |  |  |  +--ro not-a-ztp-offer?   boolean
        |  |  +--ro received-offer
        |  |  |  +--ro level?                         level
        |  |  |  +--ro not-a-ztp-offer?               boolean
        |  |  |  +--ro best?                          boolean
        |  |  |  +--ro removed-from-consideration?    boolean
        |  |  |  +--ro removal-reason?                string
        |  |  +--ro received-source-addr
        |  |  |  +--ro ipv4?   inet:ipv4-address-no-zone
        |  |  |  +--ro ipv6?   inet:ipv6-address-no-zone
        |  |  +--ro link-id-pair* [remote-id]
        |  |  |  +--ro local-id?           uint32
        |  |  |  +--ro remote-id           uint32
        |  |  |  +--ro if-index?           uint32
        |  |  |  +--ro if-name?            if:interface-ref
        |  |  |  +--ro address-families*
        |  |  |          iana-rt-types:address-family
        |  |  +--ro cost?                         uint32
        |  |  +--ro bandwidth?                    uint32
        |  |  +--ro received-link-capabilities
        |  |  |  +--ro bfd-capable?             boolean
        |  |  |  +--ro v4-forwarding-capable?   boolean
        |  |  |  +--ro mtu-size?                uint32
        |  |  +--ro received-in-lies
        |  |  |  +--ro label?                         uint32
        |  |  |  |       {label-switching}?
```

```
        |  |  |  +--ro you-are-flood-repeater?        boolean
        |  |  |  +--ro not-a-ztp-offer?               boolean
        |  |  |  +--ro you-are-sending-too-quickly?   boolean
        |  |  +--ro nbr-flood-port?               inet:port-number
        |  |  +--ro tx-flood-port?                inet:port-number
        |  |  +--ro bfd-state?                    enumeration
        |  |  +--ro outer-security-key-id?        uint8
        |  |  +--ro local-nonce?                  uint16
        |  |  +--ro remote-nonce?                 uint16
        |  |  +---x clear-neighbor
        |  +---x clear-all-neighbors
        +--ro statistics
        |  +--ro global
        |  |  +--ro total-num-routes-north?
        |  |  |       yang:zero-based-counter32
        |  |  +--ro total-num-routes-south?
        |  |          yang:zero-based-counter32
        |  +--ro spf-statistics* [spf-direction-type]
        |  |  +--ro spf-direction-type      enumeration
        |  |  +--ro start-time?             yang:date-and-time
        |  |  +--ro end-time?               yang:date-and-time
        |  |  +--ro triggering-tie
        |  |  |  +--ro tie-direction-type?     enumeration
        |  |  |  +--ro originator?             system-id
        |  |  |  +--ro tie-type?               enumeration
        |  |  |  +--ro tie-number?             uint32
        |  |  |  +--ro seq?                    uint64
        |  |  |  +--ro size?                   uint32
        |  |  |  +--ro origination-time?       ieee802-1as-timestamp
        |  |  |  +--ro origination-lifetime?   uint32
        |  |  |  +--ro remaining-lifetime?     uint32
        |  |  +---x clear-spf-statistics
        |  +--ro interfaces* [name]
        |  |  +--ro name                          if:interface-ref
        |  |  +--ro intf-states-statistics
        |  |  |  +--ro intf-states-startup-time?   uint64
        |  |  |  +--ro num-of-nbrs-3way?
        |  |  |  |       yang:zero-based-counter32
        |  |  |  +--ro num-of-nbrs-down?
        |  |  |  |       yang:zero-based-counter32
        |  |  |  +--ro nbrs-down-reasons* [system-id]
        |  |  |  |  +--ro system-id           system-id
        |  |  |  |  +--ro last-down-reason?   string
        |  |  |  +--ro num-local-level-change?
        |  |  |  |       yang:zero-based-counter32
        |  |  |  +--ro number-of-flaps?
        |  |  |  |       yang:zero-based-counter32
        |  |  |  +--ro last-state-change?          yang:date-and-time
        |  |  |  +--ro last-up?                    yang:date-and-time
        |  |  |  +--ro last-down?                  yang:date-and-time
        |  |  |  +--ro intf-lie-states
        |  |  |     +--ro last-lie-sent-time?              uint64
        |  |  |     +--ro last-lie-received-time?          uint64
        |  |  |     +--ro num-lie-received?
        |  |  |     |       yang:zero-based-counter32
        |  |  |     +--ro num-lie-transmitted?
        |  |  |     |       yang:zero-based-counter32
        |  |  |     +--ro num-lie-drop-invalid-envelope?
```

```
|  |  |     |       yang:zero-based-counter32
|  |  |     +--ro num-lie-drop-invalid-nonce?
|  |  |     |       yang:zero-based-counter32
|  |  |     +--ro num-lie-corrupted?
|  |  |             yang:zero-based-counter32
|  |  +--ro flood-repeater-statistics
|  |  |  +--ro flood-repeater?                      system-id
|  |  |  +--ro num-flood-repeater-changes?
|  |  |  |       yang:zero-based-counter32
|  |  |  +--ro last-flood-repeater-change-reason?   string
|  |  +---x clear-intf-statistics
|  +--ro neighbors* [system-id]
|     +--ro system-id               system-id
|     +--ro tie-state-statistics
|     |  +--ro transmit-queue?
|     |  |       yang:zero-based-counter32
|     |  +--ro last-queued-tie
|     |  |  +--ro tie-direction-type?     enumeration
|     |  |  +--ro originator?             system-id
|     |  |  +--ro tie-type?               enumeration
|     |  |  +--ro tie-number?             uint32
|     |  |  +--ro seq?                    uint64
|     |  |  +--ro size?                   uint32
|     |  |  +--ro origination-time?
|     |  |  |       ieee802-1as-timestamp
|     |  |  +--ro origination-lifetime?   uint32
|     |  |  +--ro remaining-lifetime?     uint32
|     |  |  +--ro reason-queued?          string
|     |  +--ro num-received-ties?
|     |  |       yang:zero-based-counter32
|     |  +--ro num-transmitted-ties?
|     |  |       yang:zero-based-counter32
|     |  +--ro num-retransmitted-ties?
|     |  |       yang:zero-based-counter32
|     |  +--ro num-flood-reduced-ties?
|     |  |       yang:zero-based-counter32
|     |  +--ro num-received-tides?
|     |  |       yang:zero-based-counter32
|     |  +--ro num-transmitted-tides?
|     |  |       yang:zero-based-counter32
|     |  +--ro num-received-tires?
|     |  |       yang:zero-based-counter32
|     |  +--ro num-transmitted-tires?
|     |  |       yang:zero-based-counter32
|     |  +--ro num-request-locally?
|     |  |       yang:zero-based-counter32
|     |  +--ro num-request-remotely?
|     |  |       yang:zero-based-counter32
|     |  +--ro num-same-older-ties-received?
|     |  |       yang:zero-based-counter32
|     |  +--ro num-seq-mismatch-pkts-received?
|     |  |       yang:zero-based-counter32
|     |  +--ro last-sent-tie
|     |  |  +--ro tie-direction-type?     enumeration
|     |  |  +--ro originator?             system-id
|     |  |  +--ro tie-type?               enumeration
|     |  |  +--ro tie-number?             uint32
|     |  |  +--ro seq?                    uint64
```

```
        |     |  |  +--ro size?                   uint32
        |     |  |  +--ro origination-time?
        |     |  |  |       ieee802-1as-timestamp
        |     |  |  +--ro origination-lifetime?   uint32
        |     |  |  +--ro remaining-lifetime?     uint32
        |     |  |  +--ro last-tie-sent-time?     yang:date-and-time
        |     |  +--ro last-recv-tie
        |     |  |  +--ro tie-direction-type?     enumeration
        |     |  |  +--ro originator?             system-id
        |     |  |  +--ro tie-type?               enumeration
        |     |  |  +--ro tie-number?             uint32
        |     |  |  +--ro seq?                    uint64
        |     |  |  +--ro size?                   uint32
        |     |  |  +--ro origination-time?
        |     |  |  |       ieee802-1as-timestamp
        |     |  |  +--ro origination-lifetime?   uint32
        |     |  |  +--ro remaining-lifetime?     uint32
        |     |  |  +--ro last-tie-recv-time?     yang:date-and-time
        |     |  +--ro largest-tie
        |     |  |  +--ro largest-tie-sent
        |     |  |  |  +--ro tie-direction-type?     enumeration
        |     |  |  |  +--ro originator?             system-id
        |     |  |  |  +--ro tie-type?               enumeration
        |     |  |  |  +--ro tie-number?             uint32
        |     |  |  |  +--ro seq?                    uint64
        |     |  |  |  +--ro size?                   uint32
        |     |  |  |  +--ro origination-time?
        |     |  |  |  |       ieee802-1as-timestamp
        |     |  |  |  +--ro origination-lifetime?   uint32
        |     |  |  |  +--ro remaining-lifetime?     uint32
        |     |  |  +--ro largest-tide-sent
        |     |  |  |  +--ro tie-direction-type?     enumeration
        |     |  |  |  +--ro originator?             system-id
        |     |  |  |  +--ro tie-type?               enumeration
        |     |  |  |  +--ro tie-number?             uint32
        |     |  |  |  +--ro seq?                    uint64
        |     |  |  |  +--ro size?                   uint32
        |     |  |  |  +--ro origination-time?
        |     |  |  |  |       ieee802-1as-timestamp
        |     |  |  |  +--ro origination-lifetime?   uint32
        |     |  |  |  +--ro remaining-lifetime?     uint32
        |     |  |  +--ro largest-tire-sent
        |     |  |     +--ro tie-direction-type?     enumeration
        |     |  |     +--ro originator?             system-id
        |     |  |     +--ro tie-type?               enumeration
        |     |  |     +--ro tie-number?             uint32
        |     |  |     +--ro seq?                    uint64
        |     |  |     +--ro size?                   uint32
        |     |  |     +--ro origination-time?
        |     |  |     |       ieee802-1as-timestamp
        |     |  |     +--ro origination-lifetime?   uint32
        |     |  |     +--ro remaining-lifetime?     uint32
        |     |  +--ro num-tie-dropped
        |     |     +--ro num-tie-outer-envelope?
        |     |     |       yang:zero-based-counter32
        |     |     +--ro num-tie-inner-envelope?
        |     |     |       yang:zero-based-counter32
        |     |     +--ro num-tie-nonce?
```

```
        |     |            yang:zero-based-counter32
        |     +---x clear-nbr-statistics
        +--ro database
           +--ro ties*
                   [tie-direction-type originator tie-type tie-number]
              +--ro tie-direction-type      enumeration
              +--ro originator              system-id
              +--ro tie-type                enumeration
              +--ro tie-number              uint32
              +--ro seq?                    uint64
              +--ro size?                   uint32
              +--ro origination-time?       ieee802-1as-timestamp
              +--ro origination-lifetime?   uint32
              +--ro remaining-lifetime?     uint32
              +--ro node
              |  +--ro level?                   level
              |  +--ro neighbors* [system-id]
              |  |  +--ro node-level?                   level
              |  |  +--ro system-id                     system-id
              |  |  +--ro fabric-id?                    uint16
              |  |  +--ro pod?                          uint32
              |  |  +--ro link-id-pair* [remote-id]
              |  |  |  +--ro local-id?           uint32
              |  |  |  +--ro remote-id           uint32
              |  |  |  +--ro if-index?           uint32
              |  |  |  +--ro if-name?            if:interface-ref
              |  |  |  +--ro address-families*
              |  |  |           iana-rt-types:address-family
              |  |  +--ro cost?                         uint32
              |  |  +--ro bandwidth?                    uint32
              |  |  +--ro received-link-capabilities
              |  |     +--ro bfd-capable?             boolean
              |  |     +--ro v4-forwarding-capable?   boolean
              |  |     +--ro mtu-size?                uint32
              |  +--ro proto-minor-ver?         uint16
              |  +--ro flood-reduction?         boolean
              |  +--ro hierarchy-indications
              |  |  +--ro hierarchy-indications?   enumeration
              |  +--ro overload-flag?           boolean
              |  +--ro name?                    string
              |  +--ro pod?                     uint32
              |  +--ro startup-time?            uint64
              |  +--ro miscabled-links*         uint32
              |  +--ro same-plane-tofs*         system-id
              |  +--ro fabric-id?               uint32
              +--ro prefixes
              |  +--ro prefixes* [prefix]
              |     +--ro prefix              inet:ip-prefix
              |     +--ro tie-type?           enumeration
              |     +--ro metric?             uint32
              |     +--ro tags*               uint64
              |     +--ro monotonic-clock
              |     |  +--ro prefix-sequence-type
              |     |     +--ro timestamp
              |     |     |       ieee802-1as-timestamp
              |     |     +--ro transaction-id?   uint8
              |     +--ro loopback?           boolean
              |     +--ro directly-attached?  boolean
```

```
            |     +--ro from-link?           uint32
            |     +--ro label?               uint32
            +--ro key-value
               +--ro key?     binary
               +--ro value?   binary

notifications:
  +---n error-set
     +--ro tie-level-error
     |  +--ro rift* [name]
     |     +--ro name    string
     |     +--ro ties* [originator]
     |        +--ro tie-direction-type?     enumeration
     |        +--ro originator              system-id
     |        +--ro tie-type?               enumeration
     |        +--ro tie-number?             uint32
     |        +--ro seq?                    uint64
     |        +--ro size?                   uint32
     |        +--ro origination-time?       ieee802-1as-timestamp
     |        +--ro origination-lifetime?   uint32
     |        +--ro remaining-lifetime?     uint32
     +--ro neighbor-error
        +--ro rift* [name]
           +--ro name          string
           +--ro interfaces* [name]
              +--ro link-id?     uint32
              +--ro name         if:interface-ref
              +--ro neighbors* [system-id]
                 +--ro system-id     system-id
                 +--ro node-level?   level
```

2.4. RIFT Configuration

The RIFT configuration includes node global configuration and interface configuration. Some features can be used to enhance protocols, such as BFD [RFC5881] with flooding reduction (Section 6.3.9 of [RFC9692]).

2.5. RIFT States

The state data nodes include node, interface, neighbor, and database information.

YANG actions are defined to clear the connection of one specific neighbor on an interface, clear the connections of all neighbors on an interface, or clear some or all statistics.

2.6. Notifications

Unexpected TIE and neighbor layer errors should be notified.

3. RIFT YANG Module

This module references [RFC9692], [RFC5881], [RFC6991], [RFC8177], [RFC8294], [RFC8343], [RFC8349], [RFC8505], and [IEEE8021AS].

```
<CODE BEGINS> file "ietf-rift@2025-04-04.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
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-routing {
    prefix rt;
    reference
      "RFC 8349: A YANG Data Model for Routing Management
       (NMDA Version)";
  }
  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }
  import ietf-routing-types {
    prefix rt-types;
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }
  import iana-routing-types {
    prefix iana-rt-types;
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }
  import ietf-key-chain {
    prefix key-chain;
    reference
      "RFC 8177: YANG Data Model for Key Chains";
  }

  organization
    "IETF RIFT (Routing In Fat Trees) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/rift/>
     WG List:  <mailto:rift@ietf.org>

     Author:   Zheng (Sandy) Zhang
               <mailto:zhang.zheng@zte.com.cn>

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

     Author:   Shaowen Ma
```

```
               <mailto:mashaowen@gmail.com>

    Author:    Xufeng Liu
               <mailto:xufeng.liu.ietf@gmail.com>

    Author:    Bruno Rijsman
               <mailto:brunorijsman@gmail.com>";
  description
    "This YANG module defines the generic configuration and
     operational state for the RIFT protocol common to all
     vendor implementations.  It is intended that the module
     will be extended by vendors to define vendor-specific
     RIFT configuration parameters and policies --
     for example, route maps or route policies.

     Copyright (c) 2025 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 Revised BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 9719
     (https://www.rfc-editor.org/info/rfc9719); see the RFC itself
     for full legal notices.

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
     NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
     'MAY', and 'OPTIONAL' in this document are to be interpreted as
     described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.";

  revision 2025-04-04 {
    description
      "Initial revision.";
    reference
      "RFC 9719: YANG Data Model for Routing in Fat Trees
       (RIFT).";
  }

  /*
   * Features
   */

  feature nonce-delta-adjust {
    description
      "Support weak nonce delta adjusting that is used in
       security.";
    reference
      "RFC 9692: RIFT: Routing in Fat Trees.
       Section 6.9.";
  }

  feature label-switching {
    description
```

```
      "Support label switching for instance distinguishing.";
    reference
      "RFC 9692: RIFT: Routing in Fat Trees.
       Section 6.8.8";
  }

  feature tie-security {
    description
      "Support security function for the TIE exchange.";
    reference
      "RFC 9692: RIFT: Routing in Fat Trees.
       Section 6.9.3.";
  }

  feature link-security {
    description
      "Support security function of link.";
    reference
      "RFC 9692: RIFT: Routing in Fat Trees.
       Section 6.9.";
  }

  typedef system-id {
    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 the pattern for RIFT System IDs.
       An example of a System ID is 0021.2FFF.FEB5.6E10.";
  }

  typedef level {
    type uint8 {
      range "0 .. 24";
    }
    default "0";
    description
      "The value of node level.
       Clos and Fat Tree networks are topologically partially
       ordered graphs and 'level' denotes the set of nodes at
       the same height in such a network.
       Nodes at the top level (i.e., ToF) are at the level with
       the highest value and count down to the nodes
       at the bottom level (i.e., leaf) with the lowest value.
       In RIFT, level 0 always indicates that a node is a leaf,
       but does not have to be level 0.
       Level values can be configured manually or automatically
       derived.";
    reference
      "RFC 9692: RIFT: Routing in Fat Trees.
       Section 6.7.";
  }

  typedef ieee802-1as-timestamp {
    type uint64;
    units "seconds";
    description
```

```
      "Timestamp per IEEE802.1AS. It is advertised with prefix
       to achieve mobility.";
    reference
      "RFC 9692: RIFT: Routing in Fat Trees. Section 6.8.4.
       IEEE8021AS: Timing and Synchronization for Time-Sensitive
                   Applications in Bridged Local Area Networks";
  }

  /*
   * Identity
   */

  identity rift {
    base rt:routing-protocol;
    description
      "Identity for the RIFT routing protocol.";
    reference
      "RFC 9692: RIFT: Routing in Fat Trees";
  }

  /*
   * Groupings
   */

  grouping address-families {
    leaf-list address-families {
      type iana-rt-types:address-family;
      description
        "Indication which address families are up on the
         interface.";
    }
    description
      "Containing address families on the interface.";
  }

  grouping hierarchy-indications {
    leaf hierarchy-indications {
      type enumeration {
        enum leaf-only {
          description
            "The node will never leave the
             'bottom of the hierarchy'.
             When this value is set, the 'configured-level'
             is the minimum level value.";
        }
        enum leaf-only-and-leaf-2-leaf-procedures {
          description
            "This means leaf to leaf.
             When this value is set, the 'configured-level'
             is the minimum level value.";
        }
        enum top-of-fabric {
          description
            "The node is 'top of fabric'.
             When this value is set, the 'configured-level'
             is the maximum level value.";
        }
      }
```

```
      description
        "The hierarchy indications of this node.";
    }
    description
      "Flags indicating node configuration in case of ZTP.";
  }

  grouping node-capability {
    leaf proto-minor-ver {
      type uint16;
      description
        "Represents the minor protocol encoding schema
         version of this node.";
    }
    leaf flood-reduction {
      type boolean;
      description
        "If the value is set to 'true', it means that
         this node enables the flood reduction function.";
    }
    container hierarchy-indications {
      config false;
      description
        "The hierarchy-indications of the node.";
      uses hierarchy-indications;
    }
    description
      "The supported capabilities of this node.";
  }

  grouping tie-type {
    leaf tie-type {
      type enumeration {
        enum illegal {
          description
            "The illegal TIE.";
        }
        enum min-tie-type {
          description
            "The minimum TIE.";
        }
        enum node {
          description
            "The node TIE.";
        }
        enum prefix {
          description
            "The prefix TIE.";
        }
        enum positive-disaggregation-prefix {
          description
            "The positive disaggregation prefix TIE.";
        }
        enum negative-disaggregation-prefix {
          description
            "The negative disaggregation prefix TIE.";
        }
        enum pgp-prefix {
```

```
          description
            "The policy guide prefix TIE.";
        }
        enum key-value {
          description
            "The key value TIE.";
        }
        enum external-prefix {
          description
            "The external prefix TIE.";
        }
        enum positive-external-disaggregation-prefix {
          description
            "The positive external disaggregation prefix TIE.";
        }
        enum max-tie-type {
          description
            "The maximum TIE.";
        }
      }
      description
        "The types of TIE.";
    }
    description
      "The types of TIE.";
  }

  grouping prefix-attribute {
    leaf metric {
      type uint32;
      description
        "The metric of this prefix.";
    }
    leaf-list tags {
      type uint64;
      description
        "The tags of this prefix.";
    }
    container monotonic-clock {
      container prefix-sequence-type {
        leaf timestamp {
          type ieee802-1as-timestamp;
          mandatory true;
          description
            "The timestamp per 802.1AS can be advertised
             with the desired prefix North TIEs.";
        }
        leaf transaction-id {
          type uint8;
          description
            "As per RFC 8505, a sequence number called a
             Transaction ID (TID) with a prefix can be
             advertised.";
          reference
            "RFC 8505: Registration Extensions for IPv6 over
             Low-Power Wireless Personal Area Network (6LoWPAN)
             Neighbor Discovery";
        }
```

```
        description
          "The prefix sequence attribute that can be advertised
           for mobility.";
        reference
          "RFC 9692: RIFT: Routing in Fat Trees.
           Section 6.8.4.";
      }
      description
        "The monotonic clock for mobile addresses.";
    }
    leaf loopback {
      type boolean;
      description
        "If the value is set to 'true', it
         indicates if the interface is a node loopback.
         The node's loopback address can be injected into
         Prefix North and Prefix South TIEs for node reachability.";
      reference
        "RFC 9692: RIFT: Routing in Fat Trees.
         Section 6.4.";
    }
    leaf directly-attached {
      type boolean;
      description
        "If the value is set to 'true', it indicates that the
         prefix is directly attached, i.e., should be routed to
         even if the node is in overload.";
    }
    leaf from-link {
      type uint32;
      description
        "In case of locally originated prefixes,
         i.e., interface addresses this can describe which
         link the address belongs to.";
    }
    leaf label {
      type uint32;
      description
        "Per prefix significant label.";
      reference
        "RFC 9692: RIFT: Routing in Fat Trees";
    }
    description
      "The attributes of the prefix.";
  }

  grouping security {
    leaf security-type {
      type enumeration {
        enum public {
          description
            "When using Public Key Infrastructure (PKI),
             the public and shared key can be used to verify
             the original packet exchanged with the neighbor.";
        }
        enum private {
          description
            "When using Public Key Infrastructure (PKI),
```

```
              the private key can be used by the Security
              fingerprint originating node to create the signature.";
          }
        }
        description
          "The security type.";
        reference
          "RFC 9692: RIFT: Routing in Fat Trees.
           Section 6.9.";
      }
      leaf shared {
        type boolean;
        description
          "When using Public Key Infrastructure (PKI),
           if the key is shared.";
        reference
          "RFC 9692: RIFT: Routing in Fat Trees.
           Section 6.9.";
      }
      choice auth-key-chain {
        description
          "Key chain or explicit key parameter specification.";
        case auth-key-chain {
          leaf key-chain {
            type key-chain:key-chain-ref;
            description
              "key-chain name.";
            reference
              "RFC 8177: YANG Data Model for Key Chains";
          }
        }
        case auth-key-explicit {
          leaf key {
            type string;
            description
              "Authentication key. The length of the key may be
               dependent on the cryptographic algorithm.";
          }
          leaf crypto-algorithm {
            type identityref {
              base key-chain:crypto-algorithm;
            }
            description
              "Cryptographic algorithm associated with key.";
            reference
              "RFC 8177: YANG Data Model for Key Chains";
          }
        }
      }
      description
        "The security parameters.";
    }

    grouping base-node-info {
      leaf node-level {
        type level;
        config false;
        description
```

```
        "The level of this node.";
    }
    leaf system-id {
      type system-id;
      mandatory true;
      description
        "Each node is identified via a system-id that is 64
         bits wide.";
    }
    leaf fabric-id {
      type uint16;
      description
        "The optional id of the fabric.";
    }
    leaf pod {
      type uint32 {
        range "1..max";
      }
      description
        "The identifier of the Point of Delivery (PoD).
         A PoD is the self-contained vertical slice of a
         Clos or Fat Tree network containing normally only leaf
         nodes (level 0) and their immediate northbound
         neighbors. It communicates with nodes
         in other PoDs via the spine. Making this leaf
         unspecified indicates that the PoD is 'undefined'.";
    }
    description
      "The base information of a node.";
  } // base-node-info

  grouping link-capabilities {
    leaf bfd-capable {
      type boolean;
      default "true";
      description
        "If this value is set to 'true', it means that
         BFD function is enabled on the neighbor.";
      reference
        "RFC 5881: Bidirectional Forwarding Detection (BFD)
         for IPv4 and IPv6 (Single Hop)";
    }
    leaf v4-forwarding-capable {
      type boolean;
      default "true";
      description
        "If this value is set to 'true', it means that
         the neighbor supports v4 forwarding.";
    }
    leaf mtu-size {
      type uint32;
      default "1400";
      description
        "MTU of the link.";
    }
    description
      "The features of neighbor.";
  } // link-capabilities
```

```
  grouping addresses {
    leaf ipv4 {
      type inet:ipv4-address-no-zone;
      description
        "IPv4 address to be used.";
    }
    leaf ipv6 {
      type inet:ipv6-address-no-zone;
      description
        "IPv6 address to be used.";
    }
    description
      "IPv4 and/or IPv6 address to be used.";
  }

  grouping lie-elements {
    leaf label {
      if-feature "label-switching";
      type uint32;
      description
        "A locally significant, downstream assigned by
         the neighbor, interface-specific label that may
         be advertised in its LIEs.";
      reference
        "RFC 9692: RIFT: Routing in Fat Trees.
         Section 6.8.8.";
    }
    leaf you-are-flood-repeater {
      type boolean;
      description
        "If the neighbor on this link is flooding repeater.
         When this value is set to 'true', the value can be
         carried in exchanged packet.";
      reference
        "RFC 9692: RIFT: Routing in Fat Trees.
         Section 6.3.9.";
    }
    leaf not-a-ztp-offer {
      type boolean;
      description
        "When this value is set to 'true', the flag can be
         carried in the LIE packet. When the value received
         in the LIE from neighbor, it indicates the level on
         the LIE MUST NOT be used to derive a ZTP level by
         the receiving node.";
      reference
        "RFC 9692: RIFT: Routing in Fat Trees.
         Section 6.7.";
    }
    leaf you-are-sending-too-quickly {
      type boolean;
      description
        "Can be optionally set to indicate to neighbor that
         packet losses are seen on reception based on packet
         numbers or the rate is too high. The receiver SHOULD
         temporarily slow down flooding rates. When this value
         is set to 'true', the flag can be carried in packet.";
```

```
    }
    description
      "The elements set in the LIEs.";
  } // lie-elements

  grouping link-id-pair {
    leaf local-id {
      type uint32;
      description
        "The local-id of link connect to this neighbor.";
    }
    leaf remote-id {
      type uint32;
      description
        "The remote-id to reach this neighbor.";
    }
    leaf if-index {
      type uint32;
      description
        "The local index of this interface.";
    }
    leaf if-name {
      type if:interface-ref;
      description
        "The name of this interface.";
    }
    uses address-families;
    description
      "A pair of local and remote link-id to identify a link
       between two nodes.";
  } // link-id-pair

  grouping neighbor-node {
    list link-id-pair {
      key "remote-id";
      uses link-id-pair;
      description
        "The multiple parallel links to this neighbor.";
    }
    leaf cost {
      type uint32;
      description
        "The cost value advertised by the neighbor.";
    }
    leaf bandwidth {
      type uint32;
      units "bits";
      description
        "Total bandwidth to the neighbor, this will be
         normally sum of the bandwidths of all the
         parallel links.";
    }
    container received-link-capabilities {
      uses link-capabilities;
      description
        "The link capabilities advertised by the neighbor.";
    }
    description
```

```
      "The neighbor information indicated in node TIE.";
  } // neighbor-node

  grouping neighbor {
    leaf proto-major-ver {
      type uint8;
      description
        "Represents protocol encoding schema major version of
         this neighbor.";
    }
    leaf proto-minor-ver {
      type uint16;
      description
        "Represents protocol encoding schema minor version of
         this neighbor.";
    }
    container sent-offer {
      leaf level {
        type level;
        description
          "The level value.";
      }
      leaf not-a-ztp-offer {
        type boolean;
        description
          "If the value is set to 'true', it indicates the
           level on the LIE MUST NOT be used to derive a
           ZTP level by the neighbor.";
      }
      description
        "The level sent to the neighbor in case the neighbor
         needs to be offered.";
    }
    container received-offer {
      leaf level {
        type level;
        description
          "The level value.";
      }
      leaf not-a-ztp-offer {
        type boolean;
        description
          "If the value is set to 'true', it indicates the
           level on the received LIE MUST NOT be used to
           derive a ZTP level.";
      }
      leaf best {
        type boolean;
        description
          "If the value is set to 'true', it means that
           the level is the best level received from all
           the neighbors.";
      }
      leaf removed-from-consideration {
        type boolean;
        description
          "If the value is set to 'true', it means that
           the level value is not considered to be used.";
```

```
      }
      leaf removal-reason {
        when "../removed-from-consideration='true'" {
          description
            "The level value is not considered to be used.";
        }
        type string;
        description
          "The reason why this value is not considered to
           be used.";
      }
      description
        "The level offered to the interface from the neighbor.
         And if the level value is considered to be used.";
    }
    container received-source-addr {
      uses addresses;
      description
        "The source address of LIE and TIE packets from
         the neighbor.";
    } // received-offer
    uses neighbor-node;
    container received-in-lies {
      uses lie-elements;
      description
        "The attributes received from this neighbor.";
    }
    leaf nbr-flood-port {
      type inet:port-number;
      default "915";
      description
        "The UDP port which is used by the neighbor to flood
         TIEs.";
    }
    leaf tx-flood-port {
      type inet:port-number;
      default "915";
      description
        "The UDP port which is used by the node to flood
         TIEs to the neighbor.";
    }
    leaf bfd-state {
      type enumeration {
        enum up {
          description
            "The link is protected by established BFD session.";
        }
        enum down {
          description
            "The link is not protected by established BFD session.";
        }
      }
      description
        "The link is protected by established BFD session or not.";
    }
    leaf outer-security-key-id {
      type uint8;
      description
```

```
        "The received security key id from the neighbor.";
      reference
        "RFC 9692: RIFT: Routing in Fat Trees.
         Section 6.9.3.";
    }
    description
      "The neighbor information.";
  } // neighbor

  grouping link-direction-type {
    leaf link-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.";
        }
      }
      config false;
      description
        "The type of link.";
    }
    description
      "The type of link.";
  } // link-direction-type

  grouping tie-direction-type {
    leaf tie-direction-type {
      type enumeration {
        enum illegal {
          description
            "Illegal direction.";
        }
        enum south {
          description
            "The direction to a node one level down.";
        }
        enum north {
          description
            "The direction to a node one level up.";
        }
        enum max {
          description
```

```
            "The max value of direction.";
        }
      }
      config false;
      description
        "The direction type of TIE.";
    }
    description
      "The direction type of TIE.";
  } // tie-direction-type

  grouping spf-direction-type {
    leaf spf-direction-type {
      type enumeration {
        enum n-spf {
          description
            "A reachability calculation that is progressing
             northbound, as example SPF that is using South
             Node TIEs only. Normally it progresses a single
             hop only and installs default routes.";
        }
        enum s-spf {
          description
            "A reachability calculation that is progressing
             southbound, as example SPF that is using North
             Node TIEs only.";
        }
      }
      config false;
      description
        "The direction type of SPF calculation.";
    }
    description
      "The direction type of SPF calculation.";
  } // spf-direction-type

  grouping tie-header {
    uses tie-direction-type;
    leaf originator {
      type system-id;
      description
        "The originator's system-id of this TIE.";
    }
    uses tie-type;
    leaf tie-number {
      type uint32;
      description
        "The number of this TIE.";
    }
    leaf seq {
      type uint64;
      description
        "The sequence number of a TIE.";
      reference
        "RFC 9692: RIFT: Routing in Fat Trees.
         Section 6.3.1.";
    }
    leaf size {
```

```
      type uint32;
      description
        "The size of this TIE.";
    }
    leaf origination-time {
      type ieee802-1as-timestamp;
      description
        "Absolute timestamp when the TIE was generated.
         This can be used on fabrics with synchronized
         clock to prevent lifetime modification attacks.";
    }
    leaf origination-lifetime {
      type uint32;
      units "seconds";
      description
        "Original lifetime when the TIE was generated.
         This can be used on fabrics with synchronized clock
         to prevent lifetime modification attacks.";
    }
    leaf remaining-lifetime {
      type uint32;
      units "seconds";
      description
        "The remaining lifetime of the TIE.";
    }
    description
      "TIEs are exchanged between RIFT nodes to describe parts
       of a network such as links and address prefixes.
       This is the TIE header information.";
  } // tie-header

  /*
   * 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 when routing protocol
         instance type is 'RIFT'.";
    }
    description
      "RIFT ( Routing in Fat Trees ) YANG model.";
    list rift {
      key "name";
      leaf name {
        type string;
        description
          "The RIFT instance's name.";
      }
      container global {
        description
          "The global configuration and status of
           this RIFT protocol instance.";
        uses base-node-info;
        leaf fabric-prefix {
          type inet:ip-prefix;
```

```
          description
            "The configured fabric prefix.";
        }
        leaf fabric-prefix-advertise {
          type boolean;
          description
            "Whether the fabric-prefix can be advertised or not.
             If the value is set to 'true', it means that
             the fabric-prefix can be advertised to neighbors.";
        }
        leaf configured-level {
          type level;
          description
            "The configured level value of this node.";
        }
        container overload {
          description
            "If the overload in TIEs can be set
             and the timeout value with according type.";
          leaf overload {
            type boolean;
            description
              "If the value is set to 'true', it means that
               the overload bit in TIEs can be set.";
          }
          choice timeout-type {
            description
              "The value of timeout timer for overloading.
               This makes sense when overload is set to 'TRUE'.";
            case on-startup {
              leaf on-startup-timeout {
                type rt-types:timer-value-seconds16;
                description
                  "Node goes into overload until this timer
                   expires when starting up.";
              }
            }
            case immediate {
              leaf immediate-timeout {
                type rt-types:timer-value-seconds16;
                description
                  "Set overload and remove after the timeout
                   expired.";
              }
            }
          }
        }
        leaf proto-major-ver {
          type uint8;
          config false;
          mandatory true;
          description
            "Represents protocol encoding schema major version.";
        }
        leaf proto-minor-ver {
          type uint16;
          config false;
          mandatory true;
```

```
          description
            "Represents protocol encoding schema minor version.";
        }
        container node-capabilities {
          uses hierarchy-indications;
          leaf flood-reduction {
            type boolean;
            description
              "If the node supports flood reduction function.
               If this value is set to 'true',  it means that
               the flood reduction function is enabled.";
            reference
              "RFC 9692: RIFT: Routing in Fat Trees.
               Section 6.3.8.";
          }
          description
            "The node's capabilities.";
        }
        leaf maximum-nonce-delta {
          if-feature "nonce-delta-adjust";
          type uint8 {
            range "1..5";
          }
          description
            "The configurable valid nonce delta value used for
             security. It is used as vulnerability window.
             If the nonces in received packet exceeds the range
             indicated by this value, the packet MUST be
             discarded.";
          reference
            "RFC 9692: RIFT: Routing in Fat Trees.
             Section 6.9.4.";
        }
        leaf nonce-increasing-interval {
          type uint16;
          units "seconds";
          description
            "The configurable nonce increasing interval.";
        }
        leaf adjusted-lifetime {
          type rt-types:timer-value-seconds16;
          units "seconds";
          description
            "The adjusted lifetime may affect the TIE stability.
             Be careful to change this parameter.
             This SHOULD be prohibited less than 2*purge-lifetime.";
        }
        container rx-lie-multicast-addr {
          leaf ipv4 {
            type inet:ipv4-address;
            default "224.0.0.121";
            description
              "The configurable LIE receiving IPv4 multicast
               address.
               Different multicast addresses can be used for
               receiving and sending.";
          }
          leaf ipv6 {
```

```
            type inet:ipv6-address;
            default "ff02::a1f7";
            description
              "The configurable LIE receiving IPv6 multicast
               address.
               Different multicast addresses can be used for
               receiving and sending.";
          }
          description
            "The configurable LIE receiving IPv4/IPv6 multicast
             address.
             Different multicast addresses can be used for
             receiving and sending.";
        }
        container tx-lie-multicast-addr {
          leaf ipv4 {
            type inet:ipv4-address;
            description
              "The configurable LIE sending IPv4 multicast
               address.
               Different multicast addresses can be used for
               receiving and sending.";
          }
          leaf ipv6 {
            type inet:ipv6-address;
            description
              "The configurable LIE sending IPv6 multicast
               address.
               Different multicast addresses can be used for
               receiving and sending.";
          }
          description
            "The configurable LIE sending IPv4/IPv6 multicast
             address.
             Different multicast addresses can be used for
             receiving and sending.";
        }
        leaf lie-tx-port {
          type inet:port-number;
          default "914";
          description
            "The UDP port of LIE packet sending. The default port
             number is 914. The value can be set to other value
             associated with different RIFT instance.";
        }
        container global-link-capabilities {
          uses link-capabilities;
          description
            "The node default link capabilities. It can be
             overwritten by the configuration underneath interface
             and neighbor.";
        }
        leaf tide-generation-interval {
          type rt-types:timer-value-seconds16;
          units "seconds";
          description
            "The TIDE generation interval.";
        }
```

```
        list tie-security {
          if-feature "tie-security";
          key "security-type";
          uses security;
          description
            "The security function used for the TIE exchange.";
          reference
            "RFC 9692: RIFT: Routing in Fat Trees.
             Section 6.9.3.";
        }
        leaf inner-security-key-id {
          type uint8;
          description
            "The inner security key id for received packet
             checking.";
          reference
            "RFC 9692: RIFT: Routing in Fat Trees.
             Section 6.9.3.";
        }
        leaf algorithm-type {
          type enumeration {
            enum spf {
              description
                "The algorithm is SPF.";
            }
            enum all-path {
              description
                "The algorithm is all-path.";
            }
          }
          description
            "The possible algorithm types.";
        }
        container hal {
          config false;
          leaf hal-value {
            type level;
            description
              "The highest defined level value seen from all
               valid level offers received.";
          }
          leaf-list system-ids {
            type system-id;
            description
              "The node's system-id of the offered level comes
               from.";
          }
          description
            "The highest defined level and the offered nodes set.";
        }
        leaf-list miscabled-links {
          type uint32;
          config false;
          description
            "List of miscabled links.";
        }
        leaf hop-limit {
          type uint8 {
```

```
            range "1 | 255";
          }
          default "1";
          description
            "The IPv4 TTL or IPv6 HL used for LIE and TIE
             sending/receiving.";
        }
        leaf maximum-clock-delta {
          type ieee802-1as-timestamp;
          description
            "The maximum drift for the timestamp comparing.";
          reference
            "RFC 9692: RIFT: Routing in Fat Trees.
             Section 6.8.4.";
        }
      }
      list interfaces {
        key "name";
        leaf link-id {
          type uint32;
          config false;
          description
            "The local id of this interface.";
        }
        leaf name {
          type if:interface-ref;
          description
            "The interface's name.";
        }
        leaf cost {
          type uint32;
          description
            "The cost from this interface to the neighbor.";
        }
        leaf rx-flood-port {
          type inet:port-number;
          default "915";
          description
            "The UDP port which is used to receive flooded
             TIEs. The default port number is 915. The value
             can be set to other value associated with different
             RIFT instance.";
        }
        leaf holdtime {
          type rt-types:timer-value-seconds16;
          units "seconds";
          default "3";
          description
            "The holding time of LIE.";
        }
        uses address-families;
        container advertised-source-addr {
          uses addresses;
          description
            "The address used in the advertised LIE and TIE
             packets.";
        }
        uses link-direction-type;
```

```
        leaf broadcast-capable {
          type boolean;
          description
            "If LIE can be received by broadcast address.";
          reference
            "RFC 9692: RIFT: Routing in Fat Trees.
             Section 6.2.";
        }
        leaf allow-horizontal-link {
          type boolean;
          description
            "If horizontal link adjacency is allowed.";
        }
        container security {
          if-feature "link-security";
          uses security;
          description
            "The security function used for this interface.";
          reference
            "RFC 9692: RIFT: Routing in Fat Trees.
             Section 6.9.3.";
        }
        leaf security-checking {
          type enumeration {
            enum no-checking {
              description
                "The security envelope does not be checked.";
            }
            enum permissive {
              description
                "The security envelope checking is permissive.";
            }
            enum loose {
              description
                "The security envelope checking is loose.";
            }
            enum strict {
              description
                "The security envelope checking is strict.";
            }
          }
          description
            "The possible security checking types.
             Only one type can be set at the same time.";
        }
        leaf was-the-last-lie-accepted {
          type boolean;
          config false;
          description
            "If the value is set to 'true', it means that
             the most recently received LIE was accepted.
             If the LIE was rejected, the neighbor error
             notifications should be used to find the reason.";
        }
        leaf last-lie-reject-reason {
          type string;
          config false;
          description
```

```
            "Description for the reject reason of the last LIE.";
        }
        container advertised-in-lies {
          config false;
          uses lie-elements;
          description
            "The attributes advertised in the LIEs from
             this interface.";
        }
        container link-capabilities {
          uses link-capabilities;
          description
            "The interface's link capabilities.";
        }
        leaf state {
          type enumeration {
            enum one-way {
              description
                "The initial state.";
            }
            enum two-way {
              description
                "Valid LIE received but not a ThreeWay LIE.";
            }
            enum three-way {
              description
                "Valid ThreeWay LIE received.";
            }
            enum multiple-neighbors-wait {
              description
                "More than two neighbors found in the same link.";
            }
          }
          config false;
          mandatory true;
          description
            "The states of LIE finite state machine.";
          reference
            "RFC 9692: RIFT: Routing in Fat Trees.
             Section 6.2.1.";
        }
        list neighbors {
          key "system-id";
          config false;
          uses base-node-info;
          uses neighbor;
          leaf local-nonce {
            type uint16;
            description
              "The exchanged local nonce with this neighbor.";
          }
          leaf remote-nonce {
            type uint16;
            description
              "The exchanged remote nonce to this neighbor.";
          }
          action clear-neighbor {
            description
```

```
              "Clears the connection to the neighbor.";
          }
          description
            "The neighbor's information.";
        }
        action clear-all-neighbors {
          description
            "Clears all the connections to the neighbors
             on this interface.";
        }
        description
          "The interface information on this node.";
      } // list interface
      container statistics {
        config false;
        container global {
          leaf total-num-routes-north {
            type yang:zero-based-counter32;
            config false;
            description
              "The total number of north routes.";
          }
          leaf total-num-routes-south {
            type yang:zero-based-counter32;
            config false;
            description
              "The total number of south routes.";
          }
          description
            "The global routes number.";
        }
        list spf-statistics {
          key "spf-direction-type";
          uses spf-direction-type;
          leaf start-time {
            type yang:date-and-time;
            description
              "The last SPF calculation start time.";
          }
          leaf end-time {
            type yang:date-and-time;
            description
              "The last SPF calculation end time.";
          }
          container triggering-tie {
            uses tie-header;
            description
              "The TIE that triggered the SPF.";
          }
          action clear-spf-statistics {
            description
              "Clears the statistics of this type of
               SPF calculation.";
          }
          description
            "The statistics of SPF calculation.";
        }
        list interfaces {
```

```
          key "name";
          leaf name {
            type if:interface-ref;
            description
              "The interface's name.";
          }
          container intf-states-statistics {
            leaf intf-states-startup-time {
              type uint64;
              description
                "The states and statistics record startup time
                 of the interface.";
            }
            leaf num-of-nbrs-3way {
              type yang:zero-based-counter32;
              description
                "The number of neighbors which state is in
                 ThreeWay.";
            }
            leaf num-of-nbrs-down {
              type yang:zero-based-counter32;
              description
                "The number of neighbors which state
                 changed to down.";
            }
            list nbrs-down-reasons {
              key "system-id";
              leaf system-id {
                type system-id;
                description
                  "The system-id of neighbor.";
              }
              leaf last-down-reason {
                type string;
                description
                  "The last down reason of the neighbor.";
              }
              description
                "The down neighbors and reasons.";
            }
            leaf num-local-level-change {
              type yang:zero-based-counter32;
              description
                "The number of local level changes.";
            }
            leaf number-of-flaps {
              type yang:zero-based-counter32;
              config false;
              description
                "The number of interface state flaps.";
            }
            leaf last-state-change {
              type yang:date-and-time;
              config false;
              description
                "Time duration in the current state.";
            }
            leaf last-up {
```

```
              type yang:date-and-time;
              config false;
              description
                "The last time of up.";
            }
            leaf last-down {
              type yang:date-and-time;
              config false;
              description
                "The last time of down.";
            }
            container intf-lie-states {
              leaf last-lie-sent-time {
                type uint64;
                description
                  "The time of the last LIE sent.";
              }
              leaf last-lie-received-time {
                type uint64;
                description
                  "The time of the last LIE received.";
              }
              leaf num-lie-received {
                type yang:zero-based-counter32;
                description
                  "The number of received LIEs.";
              }
              leaf num-lie-transmitted {
                type yang:zero-based-counter32;
                description
                  "The number of transmitted LIEs.";
              }
              leaf num-lie-drop-invalid-envelope {
                type yang:zero-based-counter32;
                description
                  "The number of dropped LIEs due to
                   invalid outer envelope.";
              }
              leaf num-lie-drop-invalid-nonce {
                type yang:zero-based-counter32;
                description
                  "The number of dropped LIEs due to
                   invalid nonce.";
              }
              leaf num-lie-corrupted {
                type yang:zero-based-counter32;
                description
                  "The number of corrupted LIEs received.";
              }
              description
                "The LIE's statistics of this interface.";
            }
            description
              "The states and statistics of this interface.";
          }
          container flood-repeater-statistics {
            leaf flood-repeater {
              type system-id;
```

```
            description
              "The system-id of the current flood repeater.
               If this leaf has no value, that means the neighbor
               is not flood repeater.";
          }
          leaf num-flood-repeater-changes {
            type yang:zero-based-counter32;
            description
              "The number of flood repeater changes.";
          }
          leaf last-flood-repeater-change-reason {
            type string;
            description
              "The reason of the last flood repeater change.";
          }
          description
            "The flood repeater statistics.";
        }
        action clear-intf-statistics {
          description
            "Clears the statistics of this interface.";
        }
        description
          "The statistics of interfaces.";
      }
      list neighbors {
        key "system-id";
        leaf system-id {
          type system-id;
          description
            "The system-id of the neighbor.";
        }
        container tie-state-statistics {
          leaf transmit-queue {
            type yang:zero-based-counter32;
            description
              "The length of TIE transmit queue.";
          }
          container last-queued-tie {
            uses tie-header;
            leaf reason-queued {
              type string;
              description
                "The queued reason of the last queued TIE.";
            }
            description
              "The last queued TIE for transmit.";
          }
          leaf num-received-ties {
            type yang:zero-based-counter32;
            description
              "The number of TIEs received.";
          }
          leaf num-transmitted-ties {
            type yang:zero-based-counter32;
            description
              "The number of TIEs transmitted.";
          }
```

```
            leaf num-retransmitted-ties {
              type yang:zero-based-counter32;
              description
                "The number of TIEs retransmitted.";
            }
            leaf num-flood-reduced-ties {
              type yang:zero-based-counter32;
              description
                "The number of TIEs that were flood reduced.";
            }
            leaf num-received-tides {
              type yang:zero-based-counter32;
              description
                "The number of TIDEs received.";
            }
            leaf num-transmitted-tides {
              type yang:zero-based-counter32;
              description
                "The number of TIDEs transmitted.";
            }
            leaf num-received-tires {
              type yang:zero-based-counter32;
              description
                "The number of TIREs received.";
            }
            leaf num-transmitted-tires {
              type yang:zero-based-counter32;
              description
                "The number of TIREs transmitted.";
            }
            leaf num-request-locally {
              type yang:zero-based-counter32;
              description
                "The number of TIEs requested locally.";
            }
            leaf num-request-remotely {
              type yang:zero-based-counter32;
              description
                "The number of TIEs requested by the neighbor.";
            }
            leaf num-same-older-ties-received {
              type yang:zero-based-counter32;
              description
                "The number of times of the same or older TIE
                 has been received.";
            }
            leaf num-seq-mismatch-pkts-received {
              type yang:zero-based-counter32;
              description
                "The number of packets with sequence number
                 mismatches.";
            }
            container last-sent-tie {
              uses tie-header;
              leaf last-tie-sent-time {
                type yang:date-and-time;
                description
                  "The time of the last TIE sent.";
```

```
            }
            description
              "The information of the last sent TIE.";
          }
          container last-recv-tie {
            uses tie-header;
            leaf last-tie-recv-time {
              type yang:date-and-time;
              description
                "The time of the last TIE received.";
            }
            description
              "The information of the last received TIE.";
          }
          container largest-tie {
            container largest-tie-sent {
              uses tie-header;
              description
                "The largest TIE sent.";
            }
            container largest-tide-sent {
              uses tie-header;
              description
                "The largest TIDE sent.";
            }
            container largest-tire-sent {
              uses tie-header;
              description
                "The largest TIRE sent.";
            }
            description
              "The largest sent TIE, TIDE and TIRE.";
          }
          container num-tie-dropped {
            leaf num-tie-outer-envelope {
              type yang:zero-based-counter32;
              description
                "The total number of TIEs dropped due to
                 invalid outer envelope.";
            }
            leaf num-tie-inner-envelope {
              type yang:zero-based-counter32;
              description
                "The total number of TIEs dropped due to
                 invalid inner envelope.";
            }
            leaf num-tie-nonce {
              type yang:zero-based-counter32;
              description
                "The total number of TIEs dropped due to
                 invalid nonce.";
            }
            description
              "The total number of TIEs dropped due to
               security reasons.";
          }
          description
            "The statistics of TIE, TIDE, TIRE
```

```
              exchanging with this neighbor.";
          }
          action clear-nbr-statistics {
            description
              "Clears the statistics of this neighbor.";
          }
          description
            "The statistics of neighbors.";
        }
        description
          "The statistics collection.";
      }
      container database {
        config false;
        list ties {
          key "tie-direction-type originator tie-type tie-number";
          description
            "A list of TIEs (Topology Information Elements).";
          uses tie-header;
          container node {
            leaf level {
              type level;
              config false;
              description
                "The level of this node.";
            }
            list neighbors {
              key "system-id";
              uses base-node-info;
              uses neighbor-node;
              description
                "The node TIE information of a neighbor.";
            }
            uses node-capability;
            leaf overload-flag {
              type boolean;
              description
                "If the value is set to 'true', it means that
                 the overload bit in TIEs is set.";
            }
            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 startup-time {
              type uint64;
```

```
            description
              "Startup time of the node.";
          }
          leaf-list miscabled-links {
            type uint32;
            config false;
            description
              "List of miscabled links.";
          }
          leaf-list same-plane-tofs {
            type system-id;
            config false;
            description
              "ToFs in the same plane. Only carried by ToF.
               Multiple Node TIEs can carry disjoint sets of
               ToFs which MUST be joined to form a single
               set.";
          }
          leaf fabric-id {
            type uint32;
            config false;
            description
              "The optional ID of the Fabric configured.";
          }
          description
            "The node element information in this TIE.";
        } // node
        container prefixes {
          description
            "The prefix element information in this TIE.";
          list prefixes {
            key "prefix";
            leaf prefix {
              type inet:ip-prefix;
              description
                "The prefix information.";
            }
            uses tie-type;
            uses prefix-attribute;
            description
              "The prefix set information.";
          }
        }
        container key-value {
          leaf key {
            type binary;
            description
              "The type of key value combination.";
          }
          leaf value {
            type binary;
            description
              "The value of key value combination.";
          }
          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.";
          } // kv-store
        } // ties
        description
          "The TIEs information in database.";
      } // container database
      description
        "RIFT configuration and state data.";
    } //rift
  } //augment

  /*
   * Notifications
   */

  notification error-set {
    description
      "The errors notification of RIFT.";
    container tie-level-error {
      description
        "The TIE errors notification of RIFT.";
      list rift {
        key "name";
        leaf name {
          type string;
          description
            "The RIFT instance's name.";
        }
        list ties {
          key "originator";
          uses tie-header;
          description
            "The level is undefined in the LIEs.";
        }
        description
          "The TIE errors set.";
      }
    }
    container neighbor-error {
      description
        "The neighbor errors notification of RIFT.";
      list rift {
        key "name";
        leaf name {
          type string;
          description
            "The RIFT instance's name.";
        }
        list interfaces {
          key "name";
          leaf link-id {
            type uint32;
            description
              "The local id of this interface.";
          }
          leaf name {
            type if:interface-ref;
```

```
            description
              "The interface's name.";
          }
          list neighbors {
            key "system-id";
            leaf system-id {
              type system-id;
              description
                "Each node is identified via a system-id which is 64
                 bits wide.";
            }
            leaf node-level {
              type level;
              description
                "The level of this node.";
            }
            description
              "The level of the neighbor is wrong.";
          }
          description
            "The interfaces with wrong level neighbor.";
        }
        description
          "The RIFT instance.";
      }
    }
  }
}

<CODE ENDS>
```

4. 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 [RFC8446].

The Network Configuration Access Control Model [RFC8341] 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. These are the subtrees and data nodes and their sensitivity/vulnerability:

- /rift

Modifying the configuration may cause all the RIFT neighborships to be rebuilt. For example, changing the configuration of configured-level or system-id will lead to all the neighbor connections of this node being rebuilt. The incorrect modification of authentication, except for the broken neighbor connection, will break the connection permanently. The modification of interface will cause the neighbor state to change. In general, unauthorized modification of most RIFT configurations will pose their own set of security risks and the "Security Considerations" in the respective RFCs referenced should be consulted.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

- /rift
- /rift/global/tie-security
- /rift/interface
- /rift/interface/neighbor
- /rift/database

The exposure of the database will expose the detailed topology of the network. Network operators may consider their topologies to be sensitive confidential data.

For RIFT authentication, configuration is supported via the specification of key chains [RFC8177] or the direct specification of key and authentication algorithm. Hence, authentication configuration inherits the security considerations of [RFC8177]. This includes the considerations with respect to the local storage and handling of authentication keys.

The actual authentication key data (whether locally specified or part of a key chain) is sensitive and needs to be kept secret from unauthorized parties. Compromise of the key data would allow an attacker to forge RIFT packets that would be accepted as authentic, potentially compromising the entire domain.

Some of 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. These are the operations and their sensitivity/vulnerability:

- /rift/interface/clear-all-neighbors
- /rift/interface/neighbor/clear-neighbor
- /rift/statistics/spf-statistics/clear-spf-statistics
- /rift/statistics/interface/clear-intf-statistics
- /rift/statistics/interface/neighbors/clear-nbr-statistics

Unauthorized access to either of the above action operations can lead to the neighbor connection rebuilding or clearing of statistics on this device.

5. IANA Considerations

Per this document, IANA has registered a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration has been made:

URI: urn:ietf:params:xml:ns:yang:ietf-rift
Registrant Contact: The IESG
XML: N/A; the requested URI is an XML namespace.

One new YANG module name has been registered in the YANG Module Names registry [RFC6020] as follows:

Name: ietf-rift
Namespace: urn:ietf:params:xml:ns:yang:ietf-rift
Prefix: rift
Reference: RFC 9719

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI 10.17487/RFC5881, June 2010, <https://www.rfc-editor.org/info/rfc5881>.

[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>.

[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>.

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

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <https://www.rfc-editor.org/info/rfc8294>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[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>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery", RFC 8505, DOI 10.17487/RFC8505, November 2018, <https://www.rfc-editor.org/info/rfc8505>.

[RFC9692] Przygienda, T., Ed., Head, J., Ed., Sharma, A., Thubert, P., Rijsman, B., and D. Afanasiev, "RIFT: Routing in Fat Trees", RFC 9692, DOI 10.17487/RFC9692, April 2025, <https://www.rfc-editor.org/info/rfc9692>.

6.2. Informative References

[IEEE8021AS] IEEE, "IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks", IEEE Std 802.1AS-2011, DOI 10.1109/IEEESTD.2011.5741898, March 2011, <https://ieeexplore.ieee.org/document/5741898/>.

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

[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>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

Acknowledgments

The authors would like to thank Tony Przygienda, Jordan Head, Benchong Xu (mailto:xu.benchong@zte.com.cn), and Tom Petch for their review, valuable comments, and suggestions.

Authors' Addresses

Zheng (Sandy) Zhang
ZTE Corporation
Email: zhang.zheng@zte.com.cn

Yuehua Wei
ZTE Corporation
Email: wei.yuehua@zte.com.cn

Shaowen Ma
Google
Email: mashaowen@gmail.com

Xufeng Liu
Individual
Email: xufeng.liu.ietf@gmail.com

Bruno Rijsman
Individual
Email: brunorijsman@gmail.com