

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: December 11, 2015

L. Zheng, Ed.  
Huawei Technologies  
R. Rahman, Ed.  
Cisco Systems  
S. Pallagatti  
Juniper Networks  
M. Jethanandani  
Cisco Systems  
G. Mirsky  
Ericsson  
June 9, 2015

Yang Data Model for Bidirectional Forwarding Detection (BFD)  
draft-zheng-bfd-yang-03.txt

## Abstract

This document defines a YANG data model that can be used to configure and manage Bidirectional Forwarding Detection (BFD).

## Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 11, 2015.

Internet-Draft

BFD Yang

June 2015

## Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">1.1.</a>	Contributors . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Design of the Data Model . . . . .	<a href="#">3</a>
<a href="#">2.1.</a>	Design of configuration model . . . . .	<a href="#">4</a>
<a href="#">2.1.1.</a>	Centralized BFD configuration . . . . .	<a href="#">4</a>
<a href="#">2.1.1.1.</a>	Common BFD configuration . . . . .	<a href="#">4</a>
<a href="#">2.1.1.2.</a>	Single-hop IP . . . . .	<a href="#">5</a>
<a href="#">2.1.1.3.</a>	Multi-hop IP . . . . .	<a href="#">5</a>
<a href="#">2.1.1.4.</a>	MPLS LSP . . . . .	<a href="#">6</a>
<a href="#">2.1.1.5.</a>	Link Aggregation Group . . . . .	<a href="#">6</a>
<a href="#">2.1.1.6.</a>	Per-interface configuration . . . . .	<a href="#">6</a>
<a href="#">2.1.2.</a>	Configuration in BFD clients . . . . .	<a href="#">6</a>
<a href="#">2.2.</a>	Design of operational model . . . . .	<a href="#">7</a>
<a href="#">2.3.</a>	Notifications . . . . .	<a href="#">7</a>
<a href="#">2.4.</a>	RPC Operations . . . . .	<a href="#">8</a>
<a href="#">2.5.</a>	BFD Configuration Data Hierarchy . . . . .	<a href="#">8</a>
<a href="#">2.5.1.</a>	Centralized BFD configuration . . . . .	<a href="#">8</a>
<a href="#">2.5.2.</a>	Configuration in BFD clients . . . . .	<a href="#">10</a>
<a href="#">2.6.</a>	Operational Data Hierarchy . . . . .	<a href="#">10</a>
<a href="#">2.7.</a>	Notifications . . . . .	<a href="#">12</a>
<a href="#">2.8.</a>	Examples . . . . .	<a href="#">13</a>
<a href="#">2.9.</a>	Interaction with other YANG modules . . . . .	<a href="#">13</a>
<a href="#">2.10.</a>	BFD Yang Module . . . . .	<a href="#">13</a>
<a href="#">2.11.</a>	BFD Client Example Configuration Yang Module . . . . .	<a href="#">27</a>
<a href="#">2.12.</a>	Security Considerations . . . . .	<a href="#">28</a>
<a href="#">2.13.</a>	IANA Considerations . . . . .	<a href="#">29</a>

<a href="#">2.14.</a>	Acknowledgements	<a href="#">29</a>
<a href="#">3.</a>	References	<a href="#">29</a>
<a href="#">3.1.</a>	Normative References	<a href="#">29</a>
<a href="#">3.2.</a>	Informative References	<a href="#">30</a>
<a href="#">Appendix A.</a>	Change log	<a href="#">30</a>

<a href="#">A.1.</a>	Changes between versions -02 and -03	<a href="#">30</a>
<a href="#">A.2.</a>	Changes between versions -01 and -02	<a href="#">30</a>
<a href="#">A.3.</a>	Changes between versions -00 and -01	<a href="#">30</a>
	Authors' Addresses	<a href="#">30</a>

## [1.](#) Introduction

YANG [[RFC6020](#)] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [[RFC6241](#)]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g RESTCONF [[I-D.ietf-netconf-restconf](#)]) and encodings other than XML (e.g JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage Bidirectional Forwarding Detection (BFD)[[RFC5880](#)]. BFD is a network protocol which is used for liveness detection of arbitrary paths between systems. Some examples of different types of paths over which we have BFD:

- 1) Two systems directly connected via IP. This is known as BFD over single-hop IP [[RFC5881](#)]
- 2) Two systems connected via multiple hops [[RFC5883](#)]
- 3) Two systems connected via MPLS Label Switched Paths (LSPs) [[RFC5884](#)]
- 4) Two systems connected via a Link Aggregation Group (LAG) interface [[RFC7130](#)]

BFD typically does not operate on its own. Various control protocols, aka BFD clients, use the services provided by BFD for their own operation [[RFC5882](#)]. The obvious candidates which use BFD

are those which do not have Hellos to detect failures (e.g. static routes) and routing protocols whose Hellos do not support sub-second failure detection, e.g OSPF and IS-IS.

## [1.1.](#) Contributors

## [2.](#) Design of the Data Model

Zheng, et al.

Expires December 11, 2015

[Page 3]

---

Internet-Draft

BFD Yang

June 2015

### [2.1.](#) Design of configuration model

The configuration model consists mainly of the parameters specified in [[RFC5880](#)]. Some examples are desired minimum transmit interval, required minimum receive interval, detection multiplier etc

Some implementations have BFD configuration under the BFD client, e.g. BFD configuration is under routing applications such as OSPF, IS-IS, BGP etc. Other implementations have BFD configuration centralized, i.e outside the multiple BFD clients. In the sections below we address both approaches.

#### [2.1.1.](#) Centralized BFD configuration

The BFD data model consists of configuring BFD sessions of different types (e.g. single-hop IP, multi-hop IP etc). Since the different session types have different keys we have a list per session type, but we use a grouping to share the common configuration data between the different session types.

##### [2.1.1.1.](#) Common BFD configuration

The common BFD session configuration items are put in a grouping to be used in multiple places, these items are:

local-multiplier

This is the detection time multiplier as defined in [[RFC5880](#)].

desired-min-tx-interval

This is the Desired Min TX Interval as defined in [[RFC5880](#)].

required-min-rx-interval

This is the Required Min RX Interval as defined in [[RFC5880](#)].

demand-enabled

Set to True to enable demand mode as defined in [[RFC5880](#)].

enable-authentication

Set to True to enable BFD authentication.

authentication-algorithm

Authentication algorithm to use (if enabled).

key-chain-name

Key-chain to be used for authentication (if enabled).

#### [2.1.1.2](#). Single-hop IP

We have a list for BFD sessions over single-hop IP. The key consists of:

interface

This is the interface on which the BFD packets for this session are transmitted and received. Examples of an interface are physical media, virtual circuit, tunnel etc.

destination address

Address belonging to the peer system as per [[RFC5881](#)]

The common configuration data in [Section 2.1.1.1](#) is used for single-hop IP. On top of that common data, we also need configuration data for echo:

desired-min-echo-tx-interval

This is the minimum interval that the local system would like to use when transmitting BFD echo packets. If 0 the echo function as defined in [[RFC5880](#)] is disabled.

required-min-echo-rx-interval

The is the Required Min Echo RX Interval as defined in [\[RFC5880\]](#).

#### [2.1.1.3.](#) Multi-hop IP

We have a list for BFD sessions over multi-hop IP. The key consists of:

source address

Address belonging to the local system as per [\[RFC5883\]](#)

destination address

Address belonging to the remote system as per [\[RFC5883\]](#)

VRF name

VRF in which the BFD multi-hop session is running

The common configuration data in [Section 2.1.1.1](#) is used for multi-hop IP. On top of that common data, we also need TTL:

ttl

TTL of outgoing BFD control packets.

#### [2.1.1.4.](#) MPLS LSP

TBD

#### [2.1.1.5.](#) Link Aggregation Group

TBD

#### [2.1.1.6.](#) Per-interface configuration

For implementations which have multiplier and intervals configured under the BFD clients we still need a central location to configure authentication, demand mode etc. This can be done by configuring the following parameters per interface:

## Common parameters

The common BFD parameters listed in [Section 2.1.1.1](#)

## Echo parameters

The echo parameters listed in [Section 2.1.1.2](#)

### [2.1.2.](#) Configuration in BFD clients

When BFD is configured in BFD clients, it is highly desirable to have BFD configuration consistency between those clients. In this approach we have a grouping for BFD configuration which applications can import in their YANG module:

- This provides consistency since the same grouping is being used in all applications making use of BFD
- Since not all implementations of those BFD clients have support for BFD, we must use if-feature in the respective YANG modules

An application importing the BFD configuration grouping could do so in a hierarchical manner if it has multiple levels at which BFD configuration can be applied. In a subsequent section we provide an example of how a BFD client would use the grouping in such a way.

The configuration items are:

enabled

Set to True to enable BFD.

local-multiplier

This the detection time multiplier as defined in [[RFC5880](#)].

desired-min-tx-interval

This the Desired Min TX Interval as defined in [[RFC5880](#)].

required-min-rx-interval

This the Required Min RX Interval as defined in [[RFC5880](#)].

### [2.2.](#) Design of operational model

The operational model contains both the overall statistics of BFD

sessions running on the device and the per session operational statistics. Since BFD is used for liveness detection of arbitrary paths, there is no uniform key to identify a BFD session. e.g. a BFD single-hop IP session is uniquely identified by the combination of destination IP address and interface whereas a multihop IP session is uniquely identified by the combination of source IP address, destination IP address and VRF. For this reason, for per session operational statistics, we do not have a single list with different type BFD sessions. Instead we have a container in which we have multiple lists, where each list corresponds to one specific path type for BFD. For example we have one operational list for BFD single-hop IP, another list for BFD multi-hop IP etc. In each list, mainly three categories of operational items are shown. The fundamental information of a BFD session such as the local discriminator, remote discriminator and the capability of supporting demand detect mode are shown in the first category. A second category includes a BFD session running information, e.g. the FSM the device in and diagnostic code received. Another example is the actual transmit interval between the control packets, which may be different from the desired minimum transmit interval configured, is shown in this category. Similar examples are actual received interval between the control packets and the actual transmit interval between the echo packets. The third category contains the detailed statistics of this session, e.g. when the session went to up/down, how long it has been since the session is up/down.

### [2.3.](#) Notifications

This YANG model defines a list of notifications to inform clients of BFD with important events detected during the protocol operation. Pair of local and remote discriminator identifies a BFD session on local system. Notification also give more important details about BFD sessions e.g. new state, time in previous state, VRF and reason for BFD session state changed.

### [2.4.](#) RPC Operations



TBD

## [2.5.](#) BFD Configuration Data Hierarchy

### [2.5.1.](#) Centralized BFD configuration

The following is the centralized configuration data hierarchy:

We have a container which contains a list for each session type

We have per-interface configuration

```
module: ietf-bfd
```

```
  +---rw bfd
```

```
    +---rw bfd-cfg
```

```
      | +---rw bfd-session-cfg {bfd-centralized-session-config}?
```

```
      | | +---rw session-ip-sh* [interface dest-addr]
```

```
      | | | +---rw interface if:interface-ref
```

```
      | | | +---rw dest-addr inet:ip-address
```

```
      | | | +---rw admin-down? boolean
```

```
      | | | +---rw local-multiplier? multiplier
```

```
      | | | +---rw desired-min-tx-interval uint32
```

```
      | | | +---rw required-min-rx-interval uint32
```

```
      | | | +---rw demand-enabled? boolean
```

```
      | | | +---rw enable-authentication? boolean
```

```
      | | | +---rw authentication-parms {bfd-authentication}?
```

```
      | | | | +---rw key-chain-name? string
```

```
      | | | | +---rw algorithm? bfd-auth-algorithm
```

```
      | | | +---rw desired-min-echo-tx-interval? uint32
```

```
      | | | +---rw required-min-echo-rx-interval? uint32
```

```
      | | +---rw session-ip-mh* [vrf-name source-addr dest-addr]
```

```
      | | | +---rw vrf-name vrfName
```

```
      | | | +---rw source-addr inet:ip-address
```

```
      | | | +---rw dest-addr inet:ip-address
```

```
      | | | +---rw admin-down? boolean
```

```
      | | | +---rw local-multiplier? multiplier
```

```
      | | | +---rw desired-min-tx-interval uint32
```

```
      | | | +---rw required-min-rx-interval uint32
```

```
      | | | +---rw demand-enabled? boolean
```

```
      | | | +---rw enable-authentication? boolean
```

```
      | | | +---rw authentication-parms {bfd-authentication}?
```

```
      | | | | +---rw key-chain-name? string
```

```
      | | | | +---rw algorithm? bfd-auth-algorithm
```

```
      | | | +---rw tx-ttl? TTL
```

```
      | | | +---rw rx-ttl? TTL
```

```
      | +---rw bfd-interface-cfg* [interface] {bfd-interface-config}?
```

```
      | | +---rw interface if:interface-ref
```

```
      | | +---rw local-multiplier? multiplier
```

```
      | | +---rw desired-min-tx-interval uint32
```

```
      | | +---rw required-min-rx-interval uint32
```

```
      | | +---rw demand-enabled? boolean
```

```
      | | +---rw enable-authentication? boolean
```

```
      | | +---rw authentication-parms {bfd-authentication}?
```

```
      | | | +---rw key-chain-name? string
```

```
      | | | +---rw algorithm? bfd-auth-algorithm
```

```
      | | +---rw desired-min-echo-tx-interval? uint32
```

```
      | | +---rw required-min-echo-rx-interval? uint32
```

### [2.5.2.](#) Configuration in BFD clients

The following is the configuration data hierarchy for a hypothetical BFD client called bfd-routing-app, the BFD configuration is supported conditionally via use of if-feature.

We have a list of areas and in each area we have a list of interfaces. The BFD configuration grouping is used in a hierarchical fashion, it can be applied in "area" and "interface":

- If BFD configuration is applied under an interface, that configuration takes precedence over any BFD configuration (if any) at the area level
- If BFD configuration is applied under an "area" and none of the interfaces in that area has BFD configuration, then all interfaces belong to the "area" in question inherit the BFD configuration for the area in question.
- If the BFD client implementation supports "interface all", then all the interfaces belonging to that area will inherit the BFD configuration under "interface all". Along with this if there are specific interface configuration then specific interface will override the "interface all" parameters.

```
module: bfd-routing-app
  +--rw area* [area-id]
    +--rw area-id      uint32
    +--rw bfd-cfg
      | +--rw enabled?          boolean
      | +--rw local-multiplier? multiplier
      | +--rw desired-min-tx-interval uint32
      | +--rw required-min-rx-interval uint32
    +--rw interface* [interface]
      +--rw interface  if:interface-ref
      +--rw bfd-cfg
        +--rw enabled?          boolean
        +--rw local-multiplier? multiplier
        +--rw desired-min-tx-interval uint32
```

+++rw required-min-rx-interval     uint32

## [2.6.](#) Operational Data Hierarchy

The complete data hierarchy of BFD YANG operational model is presented below.

Zheng, et al.

Expires December 11, 2015

[Page 10]

---

Internet-Draft

BFD Yang

June 2015

```
module: ietf-bfd
  +--rw bfd
    +--ro bfd-oper
      +--ro bfd-session-statistics
        | +--ro ip-sh-session-num?   uint32
        | +--ro ip-mh-session-num?   uint32
        | +--ro total-session-num?   uint32
        | +--ro session-up-num?      uint32
        | +--ro sess-down-num?       uint32
      +--ro bfd-session-lists
        +--ro session-ip-sh* [interface dest-addr]
          | +--ro interface           if:interface-ref
          | +--ro dest-addr           inet:ip-address
          | +--ro sesssion-type?      enumeration
          | +--ro local-discriminator? discriminator
          | +--ro remote-discriminator? discriminator
          | +--ro remote-multiplier?  multiplier
          | +--ro out-interface?      if:interface-ref
          | +--ro demand-capability?  boolean
          | +--ro session-running*
          | | +--ro local-state?      state
          | | +--ro remote-state?     state
          | | +--ro local-diagnostic? diagnostic
          | | +--ro remote-diagnostic? diagnostic
          | | +--ro detect-Mode?     enumeration
          | | +--ro actual-tx-interval? uint32
          | | +--ro actual-rx-interval? uint32
          | | +--ro actual-echo-tx-interval? uint32
          | | +--ro detect-time?      uint32
          +--ro sesssion-statistics*
            | +--ro create-time?      yang:date-and-time
            | +--ro last-down-time?   yang:date-and-time
```

```

|      +--ro last-up-time?          yang:date-and-time
|      +--ro receive-pkt?          uint64
|      +--ro send-pkt?             uint64
|      +--ro down-count?           uint32
|      +--ro receive-bad-pkt?      uint64
|      +--ro send-failed-pkt?      uint64
|      +--ro short-break-count?    uint32
+--ro session-ip-mh* [vrfName source-addr dest-addr]
    +--ro vrfName                  vrfName
    +--ro source-addr              inet:ip-address
    +--ro dest-addr                inet:ip-address
    +--ro ttl?                    TTL
    +--ro sesssion-type?           enumeration
    +--ro local-discriminator?     discriminator
    +--ro remote-discriminator?    discriminator
    +--ro remote-multiplier?       multiplier

```

```

+--ro out-interface?              if:interface-ref
+--ro demand-capability?          boolean
+--ro session-running*
|   +--ro local-state?            state
|   +--ro remote-state?          state
|   +--ro local-diagnostic?       diagnostic
|   +--ro remote-diagnostic?     diagnostic
|   +--ro detect-Mode?            enumeration
|   +--ro actual-tx-interval?     uint32
|   +--ro actual-rx-interval?     uint32
|   +--ro actual-echo-tx-interval? uint32
|   +--ro detect-time?           uint32
+--ro sesssion-statistics*
    +--ro create-time?            yang:date-and-time
    +--ro last-down-time?         yang:date-and-time
    +--ro last-up-time?           yang:date-and-time
    +--ro receive-pkt?            uint64
    +--ro send-pkt?               uint64
    +--ro down-count?             uint32
    +--ro receive-bad-pkt?        uint64
    +--ro send-failed-pkt?        uint64
    +--ro short-break-count?      uint32

```

## [2.7.](#) Notifications

The BFD YANG data model defines notifications for BFD session state changes.

```
module: ietf-bfd
notifications:
  +---n bfd-singlehop-notification
  |   +--ro local-dscr?          discriminator
  |   +--ro remote-dscr?        discriminator
  |   +--ro new-state?          state
  |   +--ro state-change-reason? string
  |   +--ro time-in-previous-state? string
  |   +--ro dest-addr?          inet:ip-address
  |   +--ro interface?          if:interface-ref
  |   +--ro echo-enabled?       boolean
  +---n bfd-multihop-notification
      +--ro local-dscr?          discriminator
      +--ro remote-dscr?        discriminator
      +--ro new-state?          state
      +--ro state-change-reason? string
      +--ro time-in-previous-state? string
      +--ro dest-addr?          inet:ip-address
      +--ro vrf-name?           vrfName
      +--ro source-addr?        inet:ip-address
```

## [2.8.](#) Examples

## [2.9.](#) Interaction with other YANG modules

TBD.

## [2.10.](#) BFD Yang Module

```
<CODE BEGINS> file "ietf-bfd@2015-06-09.yang"
module ietf-bfd {
  namespace "urn:ietf:params:xml:ns:yang:ietf-bfd";
  // replace with IANA namespace when assigned
  prefix "bfd";

  import ietf-interfaces {
    prefix "if";
  }
```

```

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

import ietf-yang-types {
    prefix "yang";
}

organization "IETF BFD Working Group";

contact
    "WG Web:  <http://tools.ietf.org/wg/bfd>
    WG List:  <rtg-bfd@ietf.org>
    WG Chair: Jeff Haas
    WG Chair: Nobo Akiya
    Editor:   Lianshu Zheng and Reshad Rahman";

description
    "This module contains the YANG definition for BFD parameters as
    per RFC5880, RFC5881 and RFC5883";

revision 2015-06-09 {
    description "Initial revision.";
    reference "RFC XXXX: A YANG data model for BFD";
}

typedef discriminator {
    type uint32 {
        range 1..4294967295;
    }
}

```

```

    description "BFD discriminator";
}

typedef diagnostic {
    type enumeration {
        enum none {
            value 0;
            description "None";
        }
        enum controlExpiry {
            value 1;

```

```

        description "Control timer expiry";
    }
    enum echoFailed {
        value 2;
        description "Echo failure";
    }
    enum nborDown {
        value 3;
        description "Neighbor down";
    }
    enum fwdingReset {
        value 4;
        description "Forwarding reset";
    }
    enum pathDown {
        value 5;
        description "Path down";
    }
    enum concPathDown {
        value 6;
        description "Concatenated path down";
    }
    enum adminDown {
        value 7;
        description "Admin down";
    }
    enum reverseConcPathDown {
        value 8;
        description "Reverse concatenated path down";
    }
}
description "BFD diagnostic";
}

typedef state {
    type enumeration {
        enum adminDown {

```

```

        value 0;
        description "admindown";
    }
    enum down {

```



```

        value 1;
        description "down";
    }
    enum init {
        value 2;
        description "init";
    }
    enum up {
        value 3;
        description "up";
    }
}
description "BFD state";
}

typedef multiplier {
    type uint8 {
        range 1..255;
    }
    description "Multiplier";
}

typedef TTL {
    type uint8 {
        range 1..255;
    }
    description "Time To Live";
}

typedef bfd-auth-algorithm {
    type enumeration {
        enum simple-password {
            description
                "Simple password";
        }

        enum keyed-md5 {
            description
                "Keyed message Digest 5";
        }

        enum meticulous-keyed-md5 {
            description
                "Meticulous keyed message Digest 5";
        }
    }
}

```

```
    }

    enum keyed-sha-1 {
        description
            "Keyed secure hash algorithm (SHA1) ";
    }

    enum meticulous-keyed-sha-1 {
        description
            "Meticulous keyed secure hash algorithm (SHA1) ";
    }
}
description "Authentication algorithm";
}

typedef vrfName {
    type string;
    description "VRF Name";
}

feature bfd-centralized-session-config {
    description "BFD session centralized config supported";
}
feature bfd-interface-config {
    description "BFD per-interface config supported";
}
feature bfd-authentication {
    description "BFD authentication supported";
}

grouping bfd-grouping-common-cfg-parms {
    description "BFD grouping for common config parameters";

    leaf local-multiplier {
        type multiplier;
        default 3;
        description "Local multiplier";
    }

    leaf desired-min-tx-interval {
        type uint32;
        units microseconds;
        mandatory true;
        description
            "Desired minimum transmit interval of control packets";
    }
}
```

leaf required-min-rx-interval {

```
    type uint32;
    units microseconds;
    mandatory true;
    description
        "Required minimum receive interval of control packets";
}
```

```
leaf demand-enabled {
    type boolean;
    default false;
    description "To enable demand mode";
}
```

```
leaf enable-authentication {
    type boolean;
    default false;
    description
        "If set, the Authentication Section is present and the
        session is to be authenticated (see RFC5880 section 6.7
        for details).";
}
```

```
container authentication-parms {
    if-feature bfd-authentication;
    description "Parameters for authentication";
    leaf key-chain-name {
        type string;
        must "../algorithm" {
            error-message
                "May not be configured without algorithm";
            description "Requires algorithm";
        }
        description
            "Key chain name";
    }
    leaf algorithm {
        type bfd-auth-algorithm;
        must "../key-chain" {
            error-message
                "May not be configured without key-chain";
        }
    }
}
```

```

        description "Requires key-chain";
    }
    description "Authentication algorithm to be used";
}
}
}

```

```

grouping bfd-grouping-echo-cfg-parms {

```

```

    description "BFD grouping for echo config parameters";
    leaf desired-min-echo-tx-interval {
        type uint32;
        units microseconds;
        default 0;
        description "Desired minumum transmit interval for echo";
    }

    leaf required-min-echo-rx-interval {
        type uint32;
        units microseconds;
        default 0;
        description "Required minimum receive interval for echo";
    }
}

grouping bfd-client-base-cfg-parms {
    description
        "BFD grouping for base config parameters which could be used
        by a protocol which is a client of BFD";

    container bfd-cfg {
        description "BFD configuration";
        leaf enabled {
            type boolean;
            default false;
            description "True if BFD is enabled";
        }

        leaf local-multiplier {
            type multiplier;
            default 3;
            description "Multiplier transmitted by local system";
        }
    }
}

```

```

}

leaf desired-min-tx-interval {
    type uint32;
    units microseconds;
    mandatory true;
    description
        "Desired minimum transmit interval of control packets";
}

leaf required-min-rx-interval {
    type uint32;
    units microseconds;
    mandatory true;
    description

```

```

        "Required minimum receive interval of control packets";
    }
}
}

grouping bfd-client-full-cfg-parms {
    description
        "BFD grouping for complete config parameters which could be
        used by a protocol which is a client of BFD.";

    container bfd-cfg {
        description "BFD configuration";
        leaf enabled {
            type boolean;
            default false;
            description "True if BFD is enabled";
        }

        uses bfd-grouping-common-cfg-parms;

        uses bfd-grouping-echo-cfg-parms;
    }
}

grouping bfd-all-session {
    description "BFD session operational information";
}

```

```

leaf sesssion-type {
  type enumeration {
    enum ip-single-hop {
      value "0";
      description "IP single hop";
    }
    enum ip-multi-hop {
      value "1";
      description "IP multi hop";
    }
  }
  description
    "BFD session type, this indicates the path type that BFD is
    running on";
}
leaf local-discriminator {
  type discriminator;
  description "Local discriminator";
}
leaf remote-discriminator {
  type discriminator;
  description "Remote discriminator";
}

```

```

}
leaf remote-multiplier {
  type multiplier;
  description "Remote multiplier";
}
leaf out-interface {
  type if:interface-ref;
  description "Outgoing physical interface name";
}
leaf demand-capability{
  type boolean;
  description "Local demand mode capability";
}

list session-running {
  description "BFD session running information";
  leaf local-state {
    type state;
    description "Local state";
  }
}

```

```

}
leaf remote-state {
    type state;
    description "Remote state";
}
leaf local-diagnostic {
    type diagnostic;
    description "Local diagnostic";
}
leaf remote-diagnostic {
    type diagnostic;
    description "Remote diagnostic";
}
leaf detect-Mode {
    type enumeration {
        enum async-with-echo {
            value "0";
            description "Async with echo";
        }
        enum async-without-echo {
            value "1";
            description "Async without echo";
        }
        enum demand-with-echo {
            value "2";
            description "Demand with echo";
        }
        enum demand-without-echo {
            value "3";

```

```

        description "Demand without echo";
    }
}
description "Detect mode";
}
leaf actual-tx-interval {
    type uint32;
    units microseconds;
    description "Actual transmit interval";
}
leaf actual-rx-interval {
    type uint32;

```

```

        units microseconds;
        description "Actual receive interval";
    }
    leaf actual-echo-tx-interval {
        type uint32;
        units microseconds;
        description "Actual echo transmit interval";
    }
    leaf detect-time {
        type uint32;
        units microseconds;
        description "Detect time";
    }
}

list sesssion-statistics {
    description "BFD session statistics";

    leaf create-time {
        type yang:date-and-time;
        description
            "Time and date when session was created";
    }
    leaf last-down-time {
        type yang:date-and-time;
        description
            "Time and date of last time the session went down";
    }
    leaf last-up-time {
        type yang:date-and-time;
        description
            "Time and date of last time the session went up";
    }
    leaf receive-pkt {
        type uint64;
        description "Received Packet Count";
    }

```

```

    }
    leaf send-pkt {
        type uint64;
        description "Sent Packet Count";
    }

```



```

    leaf down-count {
        type uint32;
        description "Session Down Count";
    }
    leaf receive-bad-pkt {
        type uint64;
        description "Received bad packet count";
    }
    leaf send-failed-pkt {
        type uint64;
        description "Packet Failed to Send Count";
    }
    leaf short-break-count {
        type uint32;
        default "0";
        description "Shortbreak count";
    }
}

container bfd {
    description "BFD top-level container";
    container bfd-cfg {
        description "BFD configuration";
        container bfd-session-cfg {
            if-feature bfd-centralized-session-config;
            description "BFD session configuration";
            list session-ip-sh {
                key "interface dest-addr";
                description "List of IP single-hop sessions";
                leaf interface {
                    type if:interface-ref;
                    description
                        "Interface on which the IP single-hop session is
                        running.";
                }
                leaf dest-addr {
                    type inet:ip-address;
                    description
                        "IP address of the peer";
                }
                leaf admin-down {
                    type boolean;

```

```

        default false;
        description
            "Is the BFD session administratively down";
    }
    uses bfd-grouping-common-cfg-parms;

    uses bfd-grouping-echo-cfg-parms;
}

list session-ip-mh {
    key "vrf-name source-addr dest-addr";
    description "List of IP multi-hop sessions";

    leaf vrf-name {
        type vrfName;
        description "Routing instance";
    }
    leaf source-addr {
        type inet:ip-address;
        description
            "Local IP address";
    }
    leaf dest-addr {
        type inet:ip-address;
        description
            "IP address of the peer";
    }
    leaf admin-down {
        type boolean;
        default false;
        description
            "Is the BFD session administratively down";
    }
    uses bfd-grouping-common-cfg-parms;

    leaf tx-ttl {
        type TTL;
        default 255;
        description "TTL of outgoing BFD control packets";
    }
    leaf rx-ttl {
        type TTL;
        mandatory true;
        description
            "Minimum allowed TTL value for incoming BFD control
            packets";
    }
}

```

Internet-Draft

BFD Yang

June 2015

```
    }

    list bfd-interface-cfg {
      if-feature bfd-interface-config;
      key interface;

      description "Per-interface BFD configuration";
      leaf interface {
        type if:interface-ref;
        description "Interface";
      }
      uses bfd-grouping-common-cfg-parms;

      uses bfd-grouping-echo-cfg-parms;
    }
  }

  container bfd-oper {
    config "false";
    description "BFD operational container";
    container bfd-session-statistics {
      description "BFD session counters";
      leaf ip-sh-session-num {
        type uint32;
        description "IP single hop session number";
      }
      leaf ip-mh-session-num {
        type uint32;
        description "IP multi hop session Number";
      }
      leaf total-session-num {
        type uint32;
        description "Total session number";
      }
      leaf session-up-num {
        type uint32;
        description "Session up number";
      }
      leaf sess-down-num {
        type uint32;
```

```

        description "Session down number";
    }
}

container bfd-session-lists {
    description "Contains multiple session lists, one per type";

```

```

list session-ip-sh {
    key "interface dest-addr";
    description "BFD IP single-hop sessions";
    leaf interface {
        type if:interface-ref;
        description
            "Interface on which the BFD session is running.";
    }
    leaf dest-addr {
        type inet:ip-address;
        description "BFD peer address";
    }
}

uses bfd-all-session;
}

list session-ip-mh {
    key "vrfName source-addr dest-addr";
    description "BFD IP multi-hop sessions";
    leaf vrfName {
        type vrfName;
        description "VRF";
    }
    leaf source-addr {
        type inet:ip-address;
        description "BFD source address";
    }
    leaf dest-addr {
        type inet:ip-address;
        description "BFD peer address";
    }
    leaf ttl {
        type TTL;
        description "TTL of session";
    }
}

```

```

        uses bfd-all-session;
    }
}
}

```

```

grouping bfd-notification-params {
    description
        "This group describes common params that will be send
        as part of BFD notification";

    leaf local-discr {
        type discriminator;
    }
}

```

```

    description "BFD local discriminator";
}

leaf remote-discr {
    type discriminator;
    description "BFD remote discriminator";
}

leaf new-state {
    type state;
    description "Current BFD state";
}

leaf state-change-reason {
    type string;
    description "BFD state change reason";
}

leaf time-in-previous-state {
    type string;
    description
        "How long the BFD session was in the previous state";
}

leaf dest-addr {
    type inet:ip-address;
    description "BFD peer address";
}

```

```

}

notification bfd-singlehop-notification {
  description
    "Notitification for BFD single-hop session state change";

  uses bfd-notification-params;

  leaf interface {
    type if:interface-ref;
    description "Interface to which this BFD session belongs to";
  }

  leaf echo-enabled {
    type boolean;
    description "Was echo enabled for BFD";
  }
}

```

```

notification bfd-multihop-notification {
  description
    "Notitification for BFD multi-hop session state change";

  uses bfd-notification-params;

  leaf vrf-name {
    type vrfName;
    description "Routing instance";
  }

  leaf source-addr {
    type inet:ip-address;
    description "BFD local address";
  }
}
}
<CODE ENDS>

```

## [2.11.](#) BFD Client Example Configuration Yang Module

```

module example-bfd-routing-app {
    namespace "urn:cisco:params:xml:ns:yang:bfdroutingapp";
    prefix bfd-routing-app;

    import ietf-bfd {
        prefix "bfd";
    }

    import ietf-interfaces {
        prefix "if";
    }

    organization
        "ACME";
    contact
        "acme@acme.com";

    description
        "Testing BFD grouping (simulating a routing application)";

    revision 2015-02-14 {
        description
            "Initial revision.";
    }

    feature routing-app-bfd {

```

```

        description "BFD configuration under routing-app";
    }

    list area {
        description
            "Specify a routing area.";

        key "area-id";

        leaf area-id {
            type uint32;
        }

        uses bfd:bfd-client-base-cfg-parms {

```

```

        if-feature routing-app-bfd;
    }

    list interface {
        key "interface";
        leaf interface {
            type if:interface-ref;
        }
        uses bfd:bfd-client-base-cfg-parms {
            if-feature routing-app-bfd;
        }
    }
}
}

```

## 2.12. Security Considerations

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

The YANG module has writeable data nodes which can be used for creation of BFD sessions and modification of BFD session parameters. The system should "police" creation of BFD sessions to prevent new sessions from causing existing BFD sessions to fail. For BFD session modification, the BFD protocol has mechanisms in place which allow for in service modification.

## 2.13. IANA Considerations

The IANA is requested to as assign a new new namespace URI from the IETF XML registry.

URI:TBD



## [2.14.](#) Acknowledgements

We would also like to thank Nobo Akiya and Jeff Haas for their encouragement on this work.

## [3.](#) References

### [3.1.](#) Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", [RFC 5880](#), June 2010.
- [RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", [RFC 5881](#), June 2010.
- [RFC5882] Katz, D. and D. Ward, "Generic Application of Bidirectional Forwarding Detection (BFD)", [RFC 5882](#), June 2010.
- [RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", [RFC 5883](#), June 2010.
- [RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", [RFC 5884](#), June 2010.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", [RFC 6241](#), June 2011.

- [RFC7130] Bhatia, M., Chen, M., Boutros, S., Binderberger, M., and J. Haas, "Bidirectional Forwarding Detection (BFD) on Link Aggregation Group (LAG) Interfaces", [RFC 7130](#), February 2014.

### [3.2.](#) Informative References

- [I-D.ietf-netconf-restconf]  
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", [draft-ietf-netconf-restconf-05](#) (work in progress), June 2015.

### [Appendix A.](#) Change log

RFC Editor: Remove this section upon publication as an RFC.

#### [A.1.](#) Changes between versions -02 and -03

- o Fixed date mismatch
- o Updated authors

#### [A.2.](#) Changes between versions -01 and -02

- o Fixed errors and warnings from "pyang --ietf"
- o Added appendix for "Change log"

#### [A.3.](#) Changes between versions -00 and -01

In the YANG module section:

- o Added missing filename
- o Added missing CODE ENDS

### Authors' Addresses

Lianshu Zheng (editor)  
Huawei Technologies  
China

Email: [vero.zheng@huawei.com](mailto:vero.zheng@huawei.com)

Internet-Draft

BFD Yang

June 2015

Reshad Rahman (editor)  
Cisco Systems  
Canada

Email: rrahman@cisco.com

Santosh Pallagatti  
Juniper Networks  
India

Email: santoshpk@juniper.net

Mahesh Jethanandani  
Cisco Systems

Email: mjethanandani@gmail.com

Greg Mirsky  
Ericsson

Email: gregory.mirsky@ericsson.com

