PCE Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 12, 2015

R. Pobbathi
V. KumarS
D. Dhody, Ed.
Huawei Technologies
J. Hardwick
Metaswitch
October 9, 2014

A YANG Data Model for Path Computation Element Communications Protocol (PCEP)
draft-pkd-pce-pcep-yang-00

Abstract

This document defines a YANG data model for the management of Path Computation Element communications Protocol (PCEP) for communications between a Path Computation Client (PCC) and a Path Computation Element (PCE), or between two PCEs. The data model includes configuration data and state data (status information and counters for the collection of statistics).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on April 12, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Path Computation Element (PCE) defined in [RFC4655] is an entity that is capable of computing a network path or route based on a network graph, and applying computational constraints. A Path Computation Client (PCC) may make requests to a PCE for paths to be computed.

PCEP is the communication protocol between a PCC and PCE and is defined in [RFC5440]. PCEP interactions include path computation requests and path computation replies as well as notifications of specific states related to the use of a PCE in the context of

Multiprotocol Label Switching (MPLS) and Generalized MPLS (GMPLS) Traffic Engineering (TE).

This document defines a YANG [RFC6020] data model for the management of PCEP speakers. It is important to establish a common data model for how PCEP speakers are identified, configured, and monitored. The data model includes configuration data and state data (status information and counters for the collection of statistics).

This document contains a specification of the base PCEP YANG module, "ietf-pcep" which provides the basic PCEP [RFC5440] data model.

[Editor's Note: Further modules augmenting the data model with advanced features maybe handled in a future revision or a separate document.]

2. 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 [RFC2119].

3. Terminology and Notation

This document uses the terminology defined in [RFC4655] and [RFC5440]. In particular, it uses the following acronyms.

- Path Computation Request message (PCReq).
- Path Computation Reply message (PCRep).
- Notification message (PCNtf).
- Error message (PCErr).
- Request Parameters object (RP).

- Synchronization Vector object (SVEC).

- Explicit Route object (ERO).

This document also uses the following term defined in [I-D.ietf-pce-pcep-mib]:

- PCEP entity: a local PCEP speaker.

- PCEP peer: to refer to a remote PCEP speaker.

- PCEP speaker: where it is not necessary to distinguish between local and remote.

3.1. Tree Diagrams

A graphical representation of the complete data tree is presented in Section 5. The meaning of the symbols in these diagrams is as follows and as per [I-D.ietf-netmod-rfc6087bis]:

- Brackets "[" and "]" enclose list keys.

- Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.

- Abbreviations before data node names: "rw" means configuration (read-write), and "ro" state data (read-only).

- Symbols after data node names: "?" means an optional node and "*" denotes a "list" or "leaf-list".

- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

- Ellipsis ("...") stands for contents of subtrees that are not shown.

3.2. Prefixes in Data Node Names

In this document, names of data nodes 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]

Table 1: Prefixes and corresponding YANG modules

4. Objectives

This section describes some of the design objectives for the model:

- In case of existing implementations, it needs to map the data model defined in this document to their proprietary native data model. To facilitate such mappings, the data model should be simple.
- The data model should be suitable for new implementations to use as is.
- Mapping to the PCEP MIB Module should be clear.
- The data model should allow for static configurations of peers.
- The data model should include read-only counters in order to gather statistics for sent and received PCEP messages, received messages with errors, and messages that could not be sent due to errors.
- It should be fairly straight forward to augment the base data model for advanced PCE features.

5. The Design of PCEP Data Model

The module, "ietf-pcep", defines the basic components of a PCE speaker.

```
module ietf-pcep
 +--rw pcep
 |  +--rw entity* [addr-type addr]
 |     +--rw addr-type             inet:ip-version
 |     +--rw addr                  inet:ip-address
 |     +--rw enabled?              boolean
 |     +--rw role                  pcep-role
 |     +--rw description?          string
 |     +--rw connect-timer?        uint32
 |     +--rw connect-max-retry?    uint32
 |     +--rw init-backoff-timer?   uint32
 |     +--rw max-backoff-timer?    uint32
 |     +--rw open-wait-timer?      uint32
 |     +--rw keep-wait-timer?      uint32
 |     +--rw keep-alive-timer?     uint32
 |     +--rw dead-timer?           uint32
 |     +--rw allow-negotiation?    boolean
 |     +--rw max-keep-alive-timer? uint32
 |     +--rw max-dead-timer?       uint32
 |     +--rw min-keep-alive-timer? uint32
 |     +--rw min-dead-timer?       uint32
 |     +--rw sync-timer?           uint32
 |     +--rw request-timer?        uint32
```



```
 |     +--rw max-sessions?         uint32
 |     +--rw max-unknown-reqs?     uint32
 |     +--rw max-unknown-msgs?     uint32
 |     +--rw peers
 |        +--rw peer* [addr-type addr]
 |           +--rw addr-type    inet:ip-version
 |           +--rw addr         inet:ip-address
 |           +--rw description? string
 +--ro pcep-state
    +--ro entity* [addr-type addr]
       +--ro index?                uint32
       +--ro addr-type             inet:ip-version
       +--ro addr                  inet:ip-address
       +--ro admin-status?         pcep-admin-status
       +--ro oper-status?          pcep-admin-status
```

```
      +--ro role?                  pcep-role
      +--ro connect-timer?         uint32
      +--ro connect-max-retry?     uint32
      +--ro init-backoff-timer?    uint32
      +--ro max-backoff-timer?     uint32
      +--ro open-wait-timer?       uint32
      +--ro keep-wait-timer?       uint32
      +--ro keep-alive-timer?      uint32
      +--ro dead-timer?            uint32
      +--ro allow-negotiation?     boolean
      +--ro max-keep-alive-timer? uint32
      +--ro max-dead-timer?        uint32
      +--ro min-keep-alive-timer? uint32
      +--ro min-dead-timer?        uint32
      +--ro sync-timer?            uint32
      +--ro request-timer?         uint32
      +--ro max-sessions?          uint32
      +--ro max-unknown-reqs?      uint32
      +--ro max-unknown-msgs?      uint32
      +--ro peers
         +--ro peer* [addr-type addr]
            +--ro addr-type                 inet:ip-version
            +--ro addr                      inet:ip-address
            +--ro role?                     pcep-role
            +--ro discontinuity-time?       yang:timestamp
            +--ro initiate-session?         boolean
            +--ro session-exists?           boolean
            +--ro num-sess-setup-ok?        yang:counter32
            +--ro num-sess-setup-fail?      yang:counter32
            +--ro session-up-time?          yang:timestamp
            +--ro session-fail-time?        yang:timestamp
            +--ro session-fail-up-time?     yang:timestamp
            +--ro avg-rsp-time?             uint32
```



```
            +--ro lwm-rsp-time?             uint32
            +--ro hwm-rsp-time?             uint32
            +--ro num-pcreq-sent?           yang:counter32
            +--ro num-pcreq-rcvd?           yang:counter32
            +--ro num-pcrep-sent?           yang:counter32
            +--ro num-pcrep-rcvd?           yang:counter32
            +--ro num-pcerr-sent?           yang:counter32
            +--ro num-pcerr-rcvd?           yang:counter32
```

```
         +--ro num-pcntf-sent?           yang:counter32
         +--ro num-pcntf-rcvd?           yang:counter32
         +--ro num-keepalive-sent?       yang:counter32
         +--ro num-keepalive-rcvd?       yang:counter32
         +--ro num-unknown-rcvd?         yang:counter32
         +--ro num-corrupt-rcvd?         yang:counter32
         +--ro num-req-sent?             yang:counter32
         +--ro num-svec-sent?            yang:counter32
         +--ro num-svec-req-sent?        yang:counter32
         +--ro num-req-sent-pend-rep?    yang:counter32
         +--ro num-req-sent-ero-rcvd?    yang:counter32
         +--ro num-req-sent-nopath-rcvd? yang:counter32
         +--ro num-req-sent-cancel-rcvd? yang:counter32
         +--ro num-req-sent-error-rcvd?  yang:counter32
         +--ro num-req-sent-timeout?     yang:counter32
         +--ro num-req-sent-cancel-sent? yang:counter32
         +--ro num-req-rcvd?             yang:counter32
         +--ro num-svec-rcvd?            yang:counter32
         +--ro num-svec-req-rcvd?        yang:counter32
         +--ro num-req-rcvd-pend-rep?    yang:counter32
         +--ro num-req-rcvd-ero-sent?    yang:counter32
         +--ro num-req-rcvd-nopath-sent? yang:counter32
         +--ro num-req-rcvd-cancel-sent? yang:counter32
         +--ro num-req-rcvd-cancel-rcvd? yang:counter32
         +--ro num-rep-rcvd-unknown?     yang:counter32
         +--ro num-req-rcvd-unknown?     yang:counter32
         +--ro num-req-sent-closed?      yang:counter32
         +--ro num-req-rcvd-closed?      yang:counter32
         +--ro sessions
            +--ro session* [initiator]
               +--ro initiator                pcep-initiator
               +--ro state-last-change?       yang:timestamp
               +--ro state?                   pcep-sess-state
               +--ro connect-retry?           yang:counter32
               +--ro local-id?                uint32
               +--ro remote-id?               uint32
               +--ro keepalive-timer?         uint32
               +--ro peer-keepalive-timer?    uint32
               +--ro dead-timer?              uint32
               +--ro peer-dead-timer?         uint32
```



```
               +--ro ka-hold-time-rem?        uint32
```

```
+--ro overloaded?                boolean
+--ro overload-time?             uint32
+--ro peer-overloaded?           boolean
+--ro peer-overload-time?        uint32
+--ro discontinuity-time?        yang:timestamp
+--ro avg-rsp-time?              uint32
+--ro lwm-rsp-time?              uint32
+--ro hwm-rsp-time?              uint32
+--ro num-pcreq-sent?            yang:counter32
+--ro num-pcreq-rcvd?            yang:counter32
+--ro num-pcrep-sent?            yang:counter32
+--ro num-pcrep-rcvd?            yang:counter32
+--ro num-pcerr-sent?            yang:counter32
+--ro num-pcerr-rcvd?            yang:counter32
+--ro num-pcntf-sent?            yang:counter32
+--ro num-pcntf-rcvd?            yang:counter32
+--ro num-keepalive-sent?        yang:counter32
+--ro num-keepalive-rcvd?        yang:counter32
+--ro num-unknown-rcvd?          yang:counter32
+--ro num-corrupt-rcvd?          yang:counter32
+--ro num-req-sent?              yang:counter32
+--ro num-svec-sent?             yang:counter32
+--ro num-svec-req-sent?         yang:counter32
+--ro num-req-sent-pend-rep?     yang:counter32
+--ro num-req-sent-ero-rcvd?     yang:counter32
+--ro num-req-sent-nopath-rcvd?  yang:counter32
+--ro num-req-sent-cancel-rcvd?  yang:counter32
+--ro num-req-sent-error-rcvd?   yang:counter32
+--ro num-req-sent-timeout?      yang:counter32
+--ro num-req-sent-cancel-sent?  yang:counter32
+--ro num-req-rcvd?              yang:counter32
+--ro num-svec-rcvd?             yang:counter32
+--ro num-svec-req-rcvd?         yang:counter32
+--ro num-req-rcvd-pend-rep?     yang:counter32
+--ro num-req-rcvd-ero-sent?     yang:counter32
+--ro num-req-rcvd-nopath-sent?  yang:counter32
+--ro num-req-rcvd-cancel-sent?  yang:counter32
+--ro num-req-rcvd-cancel-rcvd?  yang:counter32
+--ro num-rep-rcvd-unknown?      yang:counter32
+--ro num-req-rcvd-unknown?      yang:counter32
```

5.1. The Entity Lists

The PCEP yang module may contain status information for multiple logical local PCEP entities. There are several scenarios in which there may be more than one local PCEP entity, it is listed in [I-D.ietf-pce-pcep-mib].

The data model for PCEP presented in this document uses a flat list of entities. Each entity in the list is identified by its IP address (addr-type, addr). Furthermore, each entity has a mandatory "role" leaf (the local entity PCEP role). The ietf-inet-types [RFC6991] is used.

There is one list for configuration ("/pcep/entity"), and a separate list for the operational state of all entities ("/pcep-state/entity").

The PCEP MIB module [I-D.ietf-pce-pcep-mib] uses a system generated entity index as a primary index to the read only entity table. If the device implements the PCEP MIB, the "index" leaf MUST contain the value of the corresponding pcePcepEntityIndex.

5.2. The Peer Lists

The peer list contains peer that the local PCEP entity knows about. A PCEP speaker is identified by its IP address. If there is a PCEP speaker in the network that uses multiple IP addresses then it looks like multiple distinct peers to the other PCEP speakers in the network.

Since PCEP sessions can be ephemeral, the peer list tracks a peer even when no PCEP session currently exists to that peer. The statistics contained are an aggregate of the statistics for all successive sessions to that peer.

To limit the quantity of information that is stored, an implementation MAY choose to discard this information if and only if no PCEP session exists to the corresponding peer.

The data model for PCEP peer presented in this document uses a flat list of peers. Each peer in the list is identified by its IP address (addr-type, addr).

There is one list for static peer configuration ("/pcep/entity/peers"), and a separate list for the operational state of all peers (i.e. static as well as discovered)("/pcep-state/entity/peers").

5.3. The Session Lists

The session list contains PCEP session that the PCEP entity (PCE or PCC) is currently participating in. The statistics in session are semantically different from those in peer since the former apply to the current session only, whereas the latter are the aggregate for

all sessions that have existed to that peer.

Although [RFC5440] forbids there from being more than one active PCEP session between a given pair of PCEP entities at any one time, there is a window during session establishment where two sessions may exist for a given peer, one representing a session initiated by the local PCEP entity and one representing a session initiated by the peer. If either of these sessions reaches active state, then the other is discarded.

The data model for PCEP session presented in this document uses a flat list of sessions. Each session in the list is identified by its initiator. This index allows two sessions to exist transiently for a given peer, as discussed above.

There is only one list for the operational state of all sessions ("/pcep-state/entity/peers/peer/sessions/session").

6. Advanced PCE Features

This document contains a specification of the base PCEP YANG module, "ietf-pcep" which provides the basic PCEP [RFC5440] data model.

A means and procedure to handle to following PCE features needs to be decided:

- Capability and Scope
- Domain information (local/neighbour)
- Path-Key
- OF
- GCO
- P2MP
- GMPLS
- Inter-Layer

- o Stateful PCE

7. PCEP YANG Module

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number and all occurrences of the revision date below with the date of RFC publication (and remove this note).


```
<CODE BEGINS> file "ietf-pcep@2014-10-09.yang"

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

    import ietf-inet-types {
        prefix inet;
    }

    import ietf-yang-types {
        prefix yang;
    }

    organization
        "IETF PCE (Path Computation Element) Working Group";

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

         WG Chair: JP Vasseur
                   <mailto:jpv@cisco.com>

         WG Chair: Julien Meuric
                   <mailto:julien.meuric@orange.com>

         Editor:   Dhruv Dhody
                   <mailto:dhruv.ietf@gmail.com>";

    description
        "The YANG module defines a generic configuration and
```

```
         operational model for PCEP common across all of the
         vendor implementations.";


    revision 2014-10-09 {
        description "Initial revision.";
        reference
            "RFC XXXX:  A YANG Data Model for Path Computation
                        Element Communications Protocol
                        (PCEP)";
    }

    /*
     * Identities
     */
```



```
    identity pcep {
        description "Identity for the PCEP protocol.";
    }

    /*
     * Typedefs
     */
    typedef pcep-role {
        type enumeration {
            enum unknown {
                value "0";
                description
                "An unknown role";
            }
            enum pcc {
                value "1";
                description
                "The role of a Path Computation Client";
            }
            enum pce {
                value "2";
                description
                "The role of Path Computation Element";
            }
            enum pcc-and-pce {
```

```
                value "3";
                description
                "The role of both Path Computation Client and
                 Path Computation Element";
            }
        }

        description
            "The role of a PCEP speaker.
             Takes one of the following values
             - unknown(0): the role is not known.
             - pcc(1): the role is of a Path Computation
               Client (PCC).
             - pce(2): the role is of a Path Computation
               Server (PCE).
             - pccAndPce(3): the role is of both a PCC and
               a PCE.";

    }

    typedef pcep-admin-status {
        type enumeration {
                enum admin-status-up {
```



```
                value "1";
                description
                "Admin Status is Up";
            }
            enum admin-status-down {
                value "2";
                description
                "Admin Status is Down";
            }
        }

        description
        "The Admin Status of the PCEP entity.
         Takes one of the following values
             - admin-status-up(1): Admin Status is Up.
             - admin-status-down(2): Admin Status is Down";
    }
```

```
        typedef pcep-oper-status {
            type enumeration {
                enum oper-status-up {
                    value "1";
                    description
                    "The PCEP entity is active";
                }
                enum oper-status-down {
                    value "2";
                    description
                    "The PCEP entity is inactive";
                }
                enum oper-status-going-up {
                    value "3";
                    description
                    "The PCEP entity is activating";
                }
                enum oper-status-going-down {
                    value "4";
                    description
                    "The PCEP entity is deactivating";
                }
                enum oper-status-failed {
                    value "5";
                    description
                    "The PCEP entity has failed and will recover
                     when possible.";
                }
                enum oper-status-failed-perm {
                    value "6";
```



```
                    description
                    "The PCEP entity has failed and will not recover
                     without operator intervention";
                }
            }
            description
            "The operational status of the PCEP entity.
             Takes one of the following values
                 - oper-status-up(1): Active
                 - oper-status-down(2): Inactive
                 - oper-status-going-up(3): Activating
```

```
          - oper-status-going-down(4): Deactivating
          - oper-status-failed(5): Failed
          - oper-status-failed-perm(6): Failed Permanantly";
    }

    typedef pcep-initiator {
        type enumeration {
            enum local {
                value "1";
                description
                "The local PCEP entity initiated the session";
            }

            enum remote {
                value "2";
                description
                "The remote PCEP peer initiated the session";
            }
        }
        description
        "The initiator of the session, that is, whether the TCP
         connection was initiated by the local PCEP entity or
         the remote peer.
         Takes one of the following values
             - local(1): Initiated locally
             - remote(2): Initiated remotely";
    }

    typedef pcep-sess-state {
        type enumeration {
            enum tcp-pending {
                value "1";
                description
                    "The tcp-pending state of PCEP session.";
            }

            enum open-wait {
```



```
                value "2";
                description
                    "The open-wait state of PCEP session.";
            }
```

```
            enum keep-wait {
                value "3";
                description
                    "The keep-wait state of PCEP session.";
            }

            enum session-up {
                value "4";
                description
                    "The session-up state of PCEP session.";
            }
        }
        description
            "The current state of the session.

             The set of possible states excludes the idle state
             since entries do not exist in the idle state.
             Takes one of the following values
                - tcp-pending(1): PCEP TCP Pending state
                - open-wait(2): PCEP Open Wait state
                - keep-wait(3): PCEP Keep Wait state
                - session-up(4): PCEP Session Up state";
    }


    /*
     * Features - TBD
     */

    /*
     * Groupings
     */
    grouping  pcep-entity-identifier{
        description
            "This grouping defines the identifier for PCEP entity.";
        leaf addr-type {
            type inet:ip-version;
            description
                "The type of the PCEP entity's Internet address.
                 This object specifies how the value of the
                 addr leaf should be interpreted.";
        }
```

```
            leaf addr {
                type inet:ip-address;
                description
                    "The local Internet address of this PCEP entity.
                     The type is given by addr-type.

                     If operating as a PCE server, the PCEP entity
                     listens on this address.

                     If operating as a PCC, the PCEP entity binds
                     outgoing TCP connections to this address.

                     It is possible for the PCEP entity to operate
                     both as a PCC and a PCE Server, in which case it
                     uses this address both to listen for incoming
                     TCP connections and to bind outgoing TCP
                     connections.";
            }
        }

        grouping  pcep-peer-identifier{
            description
                "This grouping defines the identifier for PCEP peer.";
            leaf addr-type {
                type inet:ip-version;
                description
                    "The type of the PCEP peer's Internet address.
                     This object specifies how the value of the
                     addr leaf should be interpreted.";
            }

            leaf addr {
                type inet:ip-address;
                description
                    "The local Internet address of this PCEP peer.
                     The type is given by addr-type.";
            }
        }


        grouping pcep-entity-info{
            description
                "This grouping defines the attributes for PCEP entity.";
            leaf connect-timer {
                type uint32 {
                    range "1..65535";
                }
```

```
            units "seconds";
```



```
            default 60;
            description
                "The time in seconds that the PCEP entity will wait
                 to establish a TCP connection with a peer.  If a
                 TCP connection is not established within this time
                 then PCEP aborts the session setup attempt.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf connect-max-retry {
            type uint32;
            default 5;
            description
                "The maximum number of times the system tries to
                 establish a TCP connection to a peer before the
                 session with the peer transitions to the idle
                 state.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf init-backoff-timer {
            type uint32 {
                range "1..65535";
            }
            units "seconds";
            description
               "The initial back-off time in seconds for retrying
                a failed session setup attempt to a peer.

                The back-off time increases for each failed
                session setup attempt, until a maximum back-off
                time is reached.  The maximum back-off time is
                max-backoff-timer.";
        }

        leaf max-backoff-timer {
```

```
            type uint32;
            units "seconds";
            description
               "The maximum back-off time in seconds for retrying
                a failed session setup attempt to a peer.

                The back-off time increases for each failed session
                setup attempt, until this maximum value is reached.
```



```
                Session setup attempts then repeat periodically
                without any further increase in back-off time.";
        }

        leaf open-wait-timer {
            type uint32 {
                range "1..65535";
            }
            units "seconds";
            default 60;
            description
               "The time in seconds that the PCEP entity will wait
                to receive an Open message from a peer after the
                TCP connection has come up.

                If no Open message is received within this time then
                PCEP terminates the TCP connection and deletes the
                associated sessions.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf keep-wait-timer {
            type uint32 {
                range "1..65535";
            }
            units "seconds";
            default 60;
            description
                "The time in seconds that the PCEP entity will wait
                 to receive a Keepalive or PCErr message from a peer
                 during session initialization after receiving an
```

```
                Open message.  If no Keepalive or PCErr message is
                received within this time then PCEP terminates the
                TCP connection and deletes the associated
                sessions.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf keep-alive-timer {
            type uint32 {
                range "0..255";
            }
            units "seconds";
            default 30;
```



```
            description
                "The keep alive transmission timer that this PCEP
                 entity will propose in the initial OPEN message of
                 each session it is involved in.  This is the
                 maximum time between two consecutive messages sent
                 to a peer. Zero means that the PCEP entity prefers
                 not to send Keepalives at all.

                 Note that the actual Keepalive transmission
                 intervals, in either direction of an active PCEP
                 session, are determined by negotiation between the
                 peers as specified by RFC 5440, and so may differ
                 from this configured value.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf dead-timer {
            type uint32 {
                range "0..255";
            }
            units "seconds";
            must ". >= ../keep-alive-timer" {
                description
                    "This value MUST be greater than
```

```
                    keep-alive-timer.";

            }
            default 120;
            description
                "The dead timer that this PCEP entity will propose
                 in the initial OPEN message of each session it is
                 involved in. This is the time after which a peer
                 should declare a session down if it does not
                 receive any PCEP messages. Zero suggests that the
                 peer does not run a dead timer at all." ;
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf allow-negotiation{
            type boolean;
            description
                "Whether the PCEP entity will permit negotiation of
                 session parameters.";
        }
```



```
        leaf max-keep-alive-timer{
            type uint32 {
                range "0..255";
            }
            units "seconds";
            description
                "In PCEP session parameter negotiation in seconds,
                 the maximum value that this PCEP entity will
                 accept from a peer for the interval between
                 Keepalive transmissions. Zero means that the PCEP
                 entity will allow no Keepalive transmission at
                 all." ;
        }

        leaf max-dead-timer{
            type uint32 {
                range "0..255";
            }
            units "seconds";
```

```
            description
                "In PCEP session parameter negotiation in seconds,
                 the maximum value that this PCEP entity will accept
                 from a peer for the Dead timer.  Zero means that
                 the PCEP entity will allow not running a Dead
                 timer.";
        }

        leaf min-keep-alive-timer{
            type uint32 {
                range "0..255";
            }
            units "seconds";
            description
                "In PCEP session parameter negotiation in seconds,
                 the minimum value that this PCEP entity will
                 accept for the interval between Keepalive
                 transmissions. Zero means that the PCEP entity
                 insists on no Keepalive transmission at all.";
        }

        leaf min-dead-timer{
            type uint32 {
                range "0..255";
            }
            units "seconds";
            description
                 "In PCEP session parameter negotiationin in
                  seconds, the minimum value that this PCEP entity
```



```
                  will accept for the Dead timer.  Zero means that
                  the PCEP entity insists on not running a Dead
                  timer.";
        }

        leaf sync-timer{
            type uint32 {
                range "0..65535";
            }
            units "seconds";
            default 60;
            description
```

```
                "The value of SyncTimer in seconds is used in the
                 case of synchronized path computation request
                 using the SVEC object. Consider the case where a
                 PCReq message is received by a PCE that contains
                 the SVEC object referring to M synchronized path
                 computation requests.  If after the expiration of
                 the SyncTimer all the M path computation requests
                 have not been, received a protocol error is
                 triggered and the PCE MUST cancel the whole set
                 of path computation  requests.

                 The aim of the SyncTimer is to avoid the storage
                 of unused synchronized requests should one of
                 them get lost for some reasons (for example, a
                 misbehaving PCC).

                 Zero means that the PCEP entity does not use the
                 SyncTimer.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf request-timer{
            type uint32 {
                range "1..65535";
            }
            units "seconds";
            description
                "The maximum time that the PCEP entity will wait
                 for a response to a PCReq message.";
        }

        leaf max-sessions{
            type uint32;
            description
```



```
                "Maximum number of sessions involving this PCEP
                 entity that can exist at any time.";
        }

        leaf max-unknown-reqs{
```

```
            type uint32;
            default 5;
            description
              "The maximum number of unrecognized requests and
               replies that any session on this PCEP entity is
               willing to accept per minute before terminating
               the session.

               A PCRep message contains an unrecognized reply
               if it contains an RP object whose request ID
               does not correspond to any in-progress request
               sent by this PCEP entity.

               A PCReq message contains an unrecognized request
               if it contains an RP object whose request ID is
               zero.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf max-unknown-msgs{
            type uint32;
            default 5;
            description
             "The maximum number of unknown messages that any
              session on this PCEP entity is willing to accept
              per minute before terminating the session.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }
    }//pcep-entity-info

    grouping  pcep-stats{
        description
            "This grouping defines statistics for PCEP. It is used
             for both peer and current session.";
        leaf avg-rsp-time{
            type uint32;
            units "milliseconds";
            description
              "The average response time.
```

```
        If an average response time has not been
        calculated then this leaf has the value zero.

        If role is pcc then this leaf is meaningless
        and is set to zero.";
}

leaf lwm-rsp-time{
    type uint32;
    units "milliseconds";
    description
     "The smallest (low-water mark) response time seen.

      If no responses have been received then this
      leaf has the value zero.

      If role is pcc then this leaf is meaningless
      and is set to zero.";
}

leaf hwm-rsp-time{
    type uint32;
    units "milliseconds";
    description
     "The greatest (high-water mark) response time seen.

      If no responses have been received then this object
      has the value zero.

      If role is pcc then this field is meaningless and
      is set to zero.";
}

leaf num-pcreq-sent{
    type yang:counter32;
    description
     "The number of PCReq messages sent.";
}

leaf num-pcreq-rcvd{
    type yang:counter32;
    description
      "The number of PCReq messages received.";
}

leaf num-pcrep-sent{
    type yang:counter32;
    description
```

```
            "The number of PCRep messages sent.";
        }

        leaf num-pcrep-rcvd{
            type yang:counter32;
            description
              "The number of PCRep messages received.";
        }

        leaf num-pcerr-sent{
            type yang:counter32;
            description
              "The number of PCErr messages sent.";
        }

        leaf num-pcerr-rcvd{
            type yang:counter32;
            description
              "The number of PCErr messages received.";
        }

        leaf num-pcntf-sent{
            type yang:counter32;
            description
              "The number of PCNtf messages sent.";
        }

        leaf num-pcntf-rcvd{
            type yang:counter32;
            description
              "The number of PCNtf messages received.";
        }

        leaf num-keepalive-sent{
            type yang:counter32;
            description
              "The number of Keepalive messages sent.";
        }

        leaf num-keepalive-rcvd{
            type yang:counter32;
            description
```

```
            "The number of Keepalive messages received.";
        }

        leaf num-unknown-rcvd{
            type yang:counter32;
            description
```



```
            "The number of unknown messages received.";
        }

        leaf num-corrupt-rcvd{
            type yang:counter32;
            description
              "The number of corrupted PCEP message received.";
        }

        leaf num-req-sent{
            type yang:counter32;
            description
              "The number of requests sent.  A request corresponds
               1:1 with an RP object in a PCReq message. This might
               be greater than num-pcreq-sent because multiple
               requests can be batched into a single PCReq
               message.";
        }

        leaf num-svec-sent{
            type yang:counter32;
            description
              "The number of SVEC objects sent in PCReq messages.
               An SVEC object represents a set of synchronized
               requests.";
        }

        leaf num-svec-req-sent{
            type yang:counter32;
            description
              "The number of requests sent that appeared in one
               or more SVEC objects.";
        }

        leaf num-req-sent-pend-rep{
```

```
            type yang:counter32;
            description
              "The number of requests that have been sent for
               which a response is still pending.";
          }

          leaf num-req-sent-ero-rcvd{
            type yang:counter32;
            description
               "The number of requests that have been sent for
                which a response with an ERO object was received.
                Such responses indicate that a path was
                successfully computed by the peer.";
```



```
          }

          leaf num-req-sent-nopath-rcvd{
            type yang:counter32;
            description
              "The number of requests that have been sent for
               which a response with a NO-PATH object was
               received. Such responses indicate that the peer
               could not  find a path to satisfy the
               request.";
          }

          leaf num-req-sent-cancel-rcvd{
            type yang:counter32;
            description
              "The number of requests that were cancelled with
               a PCNtf message.

               This might be different than num-pcntf-rcvd because
               not all PCNtf messages are used to cancel requests,
               and a single PCNtf message can cancel multiple
               requests.";
          }

          leaf num-req-sent-error-rcvd{
            type yang:counter32;
            description
             "The number of requests that were rejected with a
```

```
                PCErr message.

                This might be different than num-pcerr-rcvd because
                not all PCErr messages are used to reject requests,
                and a single PCErr message can reject multiple
                requests.";
            }

            leaf num-req-sent-timeout{
                type yang:counter32;
                description
                 "The number of requests that have been sent to a peer
                 and have been abandoned because the peer has taken too
                 long to respond to them.";
            }

            leaf num-req-sent-cancel-sent{
                type yang:counter32;
                description
                 "The number of requests that were sent to the peer and
```



```
                explicitly cancelled by the local PCEP entity sending
                a PCNtf.";
            }

            leaf num-req-rcvd{
                type yang:counter32;
                description
                 "The number of requests received.  A request
                  corresponds 1:1 with an RP object in a PCReq
                  message.

                  This might be greater than num-pcreq-rcvd because
                  multiple requests can be batched into a single
                  PCReq message.";
            }

            leaf num-svec-rcvd{
                type yang:counter32;
                description
                 "The number of SVEC objects received in PCReq
                  messages. An SVEC object represents a set of
```

```
            synchronized requests.";
        }

        leaf num-svec-req-rcvd{
            type yang:counter32;
            description
              "The number of requests received that appeared in one
               or more SVEC objects.";
        }

        leaf num-req-rcvd-pend-rep{
            type yang:counter32;
            description
             "The number of requests that have been received for
              which a response is still pending.";
        }

        leaf num-req-rcvd-ero-sent{
            type yang:counter32;
            description
             "The number of requests that have been received for
              which a response with an ERO object was sent.  Such
              responses indicate that a path was successfully
              computed by the local PCEP entity.";
        }

        leaf num-req-rcvd-nopath-sent{
```



```
            type yang:counter32;
            description
             "The number of requests that have been received for
              which a response with a NO-PATH object was sent. Such
              responses indicate that the local PCEP entity could
              not find a path to satisfy the request.";
        }

        leaf num-req-rcvd-cancel-sent{
            type yang:counter32;
            description
             "The number of requests received that were cancelled
              by the local PCEP entity sending a PCNtf message.
```

```
              This might be different than num-pcntf-sent because
              not all PCNtf messages are used to cancel requests,
              and a single PCNtf message can cancel multiple
              requests.";
          }

          leaf  num-req-rcvd-cancel-rcvd{
              type yang:counter32;
              description
               "The number of requests that were received from the
                peer and explicitly cancelled by the peer sending
                a PCNtf.";
          }

          leaf  num-rep-rcvd-unknown{
              type yang:counter32;
              description
                "The number of responses to unknown requests
                 received. A response to an unknown request is a
                 response whose RP object does not contain the
                 request ID of any request that is currently
                 outstanding on the session.";
          }

          leaf  num-req-rcvd-unknown{
              type yang:counter32;
              description
                "The number of unknown requests that have been
                 received. An unknown request is a request
                 whose RP object contains a request ID of
                 zero.";
          }

       }//pcep-stats
```



```
       /*
        * Configuration data nodes
        */
       container pcep{

          description
              "Parameters for list of configured PCEP entities
```

```
             on the device.";

          list entity{
              key "addr-type addr";

              description
                  "The configured PCEP entity on the device.";

              uses  pcep-entity-identifier {
                  description
                      "The configured PCEP entity Identifier.";
              }

              leaf enabled {
                  type boolean;
                  default true;
                  description
                      "The administrative status of this PCEP
                       Entity.";
              }

              leaf role {
                  type pcep-role;
                  mandatory true;
                  description
                      "The role that this entity can play.
                       Takes one of the following values.
                       - unknown(0): this PCEP Entity role is not
                         known.
                       - pcc(1): this PCEP Entity is a PCC.
                       - pce(2): this PCEP Entity is a PCE.
                       - pcc-and-pce(3): this PCEP Entity is both
                         a PCC and a PCE.";
              }

              leaf description {
                  type string;
                  description
                      "Description of the PCEP entity configured
                       by the user";
              }
```

```
        uses pcep-entity-info {
            description
                "The configuration related to the PCEP
                 entity.";
        }

        container peers{
            description
                    "The list of configured peers for the
                     entity";

            list peer{
                key "addr-type addr";

                description
                    "The peer configured for the entity.";

                 uses pcep-peer-identifier{
                    description
                        "The configured PCEP peer's
                         Identifier.";
                 }

                 leaf description {
                    type string;
                    description
                        "Description of the PCEP peer
                         configured by the user";
                }
            }//peer
        }//peers
    }//entity
}//pcep

/*
 * Operational data nodes
 */

container pcep-state{
    config false;
    description
        "The list of operational PCEP entities on the
         device.";

    list entity{
        key "addr-type addr";
        unique "index";
```

```
            description
                "The operational PCEP entity on the device.";

            leaf index{
                type uint32;
                description
                    "The index of the PECP operational
                     entity";
            }

            uses  pcep-entity-identifier {
                description
                    "The PCEP entity Identifier.";
            }

            leaf admin-status {
                type pcep-admin-status;
                description
                    "The administrative status of this PCEP Entity.
                     This is the desired operational status as
                     currently set by an operator or by default in
                     the implementation.  The value of enabled
                     represents the current status of an attempt
                     to reach this desired status.";
            }

            leaf oper-status {
                type pcep-admin-status;
                description
                   "The operational status of the PCEP entity.
                    Takes one of the following values.
                    - oper-status-up(1): the PCEP entity is
                      active.
                    - oper-status-down(2): the PCEP entity is
                      inactive.
                    - oper-status-going-up(3): the PCEP entity is
                      activating.
                    - oper-status-going-down(4): the PCEP entity is
                      deactivating.
                    - oper-status-failed(5): the PCEP entity has
                      failed and will recover when possible.
                    - oper-status-failed-perm(6): the PCEP entity
                      has failed and will not recover without
```

```
                operator intervention.";
            }

            leaf role {
                type pcep-role;
```



```
                description
                    "The role that this entity can play.
                     Takes one of the following values.
                     - unknown(0): this PCEP entity role is
                       not known.
                     - pcc(1): this PCEP entity is a PCC.
                     - pce(2): this PCEP entity is a PCE.
                     - pcc-and-pce(3): this PCEP entity is
                       both a PCC and a PCE.";
            }


            uses pcep-entity-info{
                description
                    "The operational information related to the
                     PCEP entity.";
            }

            container peers{
                description
                        "The list of peers for the entity";

                list peer{
                    key "addr-type addr";

                    description
                        "The peer for the entity.";

                     uses pcep-peer-identifier{
                        description
                            "The PCEP peer's Identifier.";
                     }

                     leaf role {
                        type pcep-role;
                        description
```

```
                    "The role of the PCEP Peer.
                     Takes one of the following values.
                     - unknown(0): this PCEP peer role
                       is not known.
                     - pcc(1): this PCEP peer is a PCC.
                     - pce(2): this PCEP peer is a PCE.
                     - pcc-and-pce(3): this PCEP peer
                       is both a PCC and a PCE.";
                }

                leaf discontinuity-time {
                    type yang:timestamp;
```



```
                    description
                        "The timestamp of the time when the
                         information and statistics were
                         last reset.";
                }

                leaf initiate-session {
                    type boolean;
                    description
                        "Indicates whether the local PCEP
                         entity initiates sessions to this peer,
                         or waits for the peer to initiate a
                         session.";
                }

                leaf session-exists{
                    type boolean;
                    description
                        "Indicates whether a session with
                         this peer currently exists.";
                }

                leaf num-sess-setup-ok{
                    type yang:counter32;
                    description
                        "The number of PCEP sessions successfully
                         successfully established with the peer,
                         including any current session.  This
                         counter is incremented each time a
```

```
                session with this peer is successfully
                established.";
            }

            leaf num-sess-setup-fail{
                type yang:counter32;
                description
                   "The number of PCEP sessions with the peer
                    that have been attempted but failed
                    before being fully established. This
                    counter is incremented each time a
                    session retry to this peer fails.";
            }

            leaf session-up-time{
                type yang:timestamp;
                description
                   "The timestamp value of the last time a
                    session with this peer was successfully
```



```
                    established.

                    If num-sess-setup-ok is zero, then this
                    leaf contains zero.";
            }

            leaf session-fail-time{
                type yang:timestamp;
                description
                  "The timestamp value of the last time a
                   session with this peer failed to be
                   established.

                   If num-sess-setup-fail is zero, then
                   this leaf contains zero.";
            }

            leaf session-fail-up-time{
                type yang:timestamp;
                description
                  "The timestamp value of the last time a
                   session with this peer failed from active.
```

```
                    If num-sess-setup-ok is zero, then this
                    leaf contains zero.";
                }

                uses pcep-stats{
                    description
                        "Since PCEP sessions can be ephemeral,
                         the peer statistics tracks a peer even
                         when no PCEP session currently exists
                         to that peer. The statistics contained
                         are an aggregate of the statistics for
                         all successive sessions to that peer.";
                }

                leaf num-req-sent-closed{
                    type yang:counter32;
                    description
                        "The number of requests that were sent
                         to the peer and implicitly cancelled
                         when the session they were sent over
                         was closed.";
                }

                leaf  num-req-rcvd-closed{
                    type yang:counter32;
```



```
                    description
                        "The number of requests that were
                         received from the peer and implicitly
                         cancelled when the session they were
                         received over was closed.";
                }

                container sessions {
                    description
                        "This entry represents a single PCEP
                         session in which the local PCEP entity
                         participates.

                         This entry exists only if the
                         corresponding PCEP session has been
```

```
                    initialized by some event, such as
                    manual user configuration, auto-
                    discovery of a peer, or an incoming
                    TCP connection.";

                list session {
                    key "initiator";

                    description
                        "The list of sessions, note that
                         for a time being two sessions
                         may exist for a peer";

                    leaf initiator {
                        type pcep-initiator;
                        description
                            "The initiator of the session,
                             that is, whether the TCP
                             connection was initiated by
                             the local PCEP entity or the
                             peer.

                             There is a window during
                             session initialization where
                             two sessions can exist between
                             a pair of PCEP speakers, each
                             initiated by one of the
                             speakers. One of these
                             sessions is always discarded
                             before it leaves OpenWait state.
                             However, before it is discarded,
                             two sessions to the given peer
                             appear transiently in this MIB
```



```
                             module. The sessions are
                             distinguished by who initiated
                             them, and so this field is the
                             key.";
                    }

                    leaf state-last-change {
                        type yang:timestamp;
```

```
                    description
                        "The timestamp value at the
                         time this session entered its
                         current state as denoted by
                         the state leaf.";
                }

                leaf state {
                    type pcep-sess-state;
                    description
                        "The current state of the
                         session.

                         The set of possible states
                         excludes the idle state since
                         entries do not exist in the
                         idle state.";
                }

                leaf connect-retry {
                    type yang:counter32;
                    description
                         "The number of times that the
                          local PCEP entity has
                          attempted to establish a TCP
                          connection for this session
                          without success. The PCEP
                          entity gives up when this
                          reaches connect-max-retry.";
                }

                leaf local-id {
                    type uint32 {
                      range "0..255";
                    }
                    description
                         "The value of the PCEP session
                          ID used by the local PCEP
                          entity in the Open message
                          for this session.
```



```
                          If state is tcp-pending then
```

```
                                        this is the session ID that
                                        will be used in the Open
                                        message. Otherwise, this is
                                        the session ID that was sent
                                        in the Open message.";
                                }

                                leaf remote-id {
                                    type uint32 {
                                      range "0..255";
                                    }
                                    description
                                         "The value of the PCEP session
                                          ID
    used by the peer in its
                                          Open message for this session.

                                          If state is tcp-pending or
                                          open-wait then this leaf is not
                                          used and MUST be set to zero.";
                                }

                                leaf keepalive-timer {
                                    type uint32 {
                                      range "0..255";
                                    }
                                    units "seconds";
                                    description
                                         "The agreed maximum interval at
                                          which the local PCEP entity
                                          transmits PCEP messages on this
                                          PCEP session.  Zero means that
                                          the local PCEP entity never
                                          sends Keepalives on this
                                          session.

                                          This field is used if and only
                                          if state is session-up.
                                          Otherwise, it is not used and
                                          MUST be set to zero.";
                                }

                                leaf peer-keepalive-timer {
                                    type uint32 {
                                      range "0..255";
                                    }
                                    units "seconds";
```

```
        description
             "The agreed maximum interval at
              which the peer transmits PCEP
              messages on this PCEP session.
              Zero means that the peer never
              sends Keepalives on this
              session.

              This field is used if and only
              if state is session-up.
              Otherwise, it is not used and
              MUST be set to zero.";
      }

      leaf dead-timer {
        type uint32 {
          range "0..255";
        }
        units "seconds";
        description
             "The dead timer interval for
              this PCEP session.";
      }

      leaf peer-dead-timer {
        type uint32 {
          range "0..255";
        }
        units "seconds";
        description
             "The peer's dead-timer interval
              for this PCEP session.

              If state is tcp-pending or
              open-wait then this leaf is
              not used and MUST be set to
              zero.";
      }

      leaf ka-hold-time-rem {
        type uint32 {
          range "0..255";
        }
        units "seconds";
        description
             "The keep alive hold time
```

```
                                remaining for this session.
```



```
                                If state is tcp-pending or
                                open-wait then this field is
                                not used and MUST be set to
                                zero.";
                        }

                        leaf overloaded {
                            type boolean;
                            description
                                 "If the local PCEP entity has
                                  informed the peer that it is
                                  currently overloaded, then this
                                  is set to true.  Otherwise, it
                                  is set to false.";
                        }

                        leaf overload-time {
                            type uint32;
                            units "seconds";
                            description
                                 "The interval of time that is
                                  remaining until the local PCEP
                                  entity will cease to be
                                  overloaded on this session.

                                  This field is only used if
                                  overloaded is set to true.
                                  Otherwise, it is not used and
                                  MUST be set to zero.";
                        }

                        leaf peer-overloaded {
                            type boolean;
                            description
                                 "If the peer has informed the
                                  local PCEP entity that it is
                                  currently overloaded, then this
                                  is set to true. Otherwise, it
                                  is set to false.";
```

```
                                }

                                leaf peer-overload-time {
                                    type uint32;
                                    units "seconds";
                                    description
                                         "The interval of time that is
                                          remaining until the peer will
                                          cease to be overloaded.  If it
```



```
                                          is not known how long the peer
                                          will stay in overloaded state,
                                          this leaf is set to zero.

                                          This field is only used if
                                          peer-overloaded is set to true.
                                          Otherwise, it is not used and
                                          MUST be set to zero.";
                                }

                                leaf discontinuity-time {
                                    type yang:timestamp;
                                    description
                                         "The timestamp value of the time
                                          when the statistics were last
                                          reset.";
                                }

                                uses pcep-stats{
                                    description
                                        "The statistics contained are
                                         for the current sessions to that
                                         peer. These are lost when the
                                         session goes down.";

                                }

                            } // session
                        } // sessions
                    }//peer
                }//peers
            }//entity
```

```
    }//pcep-state
}//module
<CODE ENDS>
/* To Do - Notifications, rpc (if any) */
```

8. Security Considerations

This document does not add any new security concerns beyond those discussed in [RFC5440].

9. Manageability Considerations

9.1. Control of Function and Policy

9.2. Information and Data Models

9.3. Liveness Detection and Monitoring

9.4. Verify Correct Operations

9.5. Requirements On Other Protocols

9.6. Impact On Network Operations

10. IANA Considerations

11. Acknowledgements

The initial document is based on the PCEP MIB [I-D.ietf-pce-pcep-mib]. Further this document structure is based on Routing Yang Module [I-D.ietf-netmod-routing-cfg]. We would like to thank the authors of aforementioned documents.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5440] Vasseur, JP. and JL. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, March 2009.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

12.2. Informative References

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, August 2006.

[I-D.ietf-pce-pcep-mib]
Sreenivasa, K., Emile, S., Zhao, Q., King, D., and J. Hardwick, "Path Computation Element Communications Protocol (PCEP) Management Information Base (MIB) Module", draft-ietf-pce-pcep-mib-10 (work in progress), September 2014.

[I-D.ietf-netmod-routing-cfg]
Lhotka, L., "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-15 (work in progress), May 2014.

[I-D.ietf-netmod-rfc6087bis]
Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-00 (work in progress), June 2014.

Appendix A. Contributor Addresses

Young Lee
Huawei Technologies
5340 Legacy Drive, Building 3
Plano, TX 75023, USA

Phone: (469) 277-5838
EMail: leeyoung@huawei.com

Udayasree Palle

Huawei Technologies
Leela Palace
Bangalore, Karnataka 560008
India

EMail: udayasree.palle@huawei.com

Vishnu Pavan Beeram
Juniper Networks
USA

EMail: vbeeram@juniper.net

Authors' Addresses

Rohit Pobbathi
Huawei Technologies
Leela Palace
Bangalore, Karnataka 560008
India

EMail: rohit.pobbathi@huawei.com

Vinod KumarS
Huawei Technologies
Leela Palace
Bangalore, Karnataka 560008
India

EMail: vinods.kumar@huawei.com

Dhruv Dhody (editor)
Huawei Technologies
Leela Palace
Bangalore, Karnataka 560008

India

EMail: dhruv.ietf@gmail.com

Jonathan Hardwick
Metaswitch
100 Church Street
Enfield EN2 6BQ
UK

EMail: jonathan.hardwick@metaswitch.com