Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 19, 2018

M. Boucadair
C. Jacquenet
Orange
S. Sivakumar
Cisco Systems
S. Vinapamula
Juniper Networks

October 16, 2017

YANG Modules for the Port Control Protocol (PCP)
draft-boucadair-pcp-yang-05

Abstract

This document defines YANG modules for the Port Control Protocol (PCP), including PCP client, PCP server, PCP proxy, and Universal Plug and Play (UPnP) Internet Gateway Device - Port Control Protocol Interworking Function.

Editorial Note (To be removed by RFC Editor)

Please update this statement with the RFC number to be assigned to this document:

"This version of this YANG module is part of RFC XXXX;"

Status of This Memo

Table of Contents

1. Introduction

This document defines a data model for the Port Control Protocol (PCP, [RFC6887]) using the YANG data modeling language [RFC7950]. The following functional elements are in scope:

- PCP client [RFC6887].

- PCP server [RFC6887].
- PCP proxy [RFC7648].
- Universal Plug and Play (UPnP) Internet Gateway Device - Port Control Protocol Interworking Function (UPnP IGD-PCP IWF) [RFC6970].

In addition to the base features defined in [RFC6887], this document covers the following capabilities:

- PCP Description option [RFC7220].
- PCP Prefix64 discovery option [RFC7225].
- PCP Port set allocation [RFC7753].

In conformance with [RFC7291] and [RFC7488], this document assumes that multiple PCP servers may be configured to a PCP client, PCP proxy, or UPnP IGD-PCP IWF; each server is defined by a list of IP addresses.

This document follows the guidelines of [RFC6087].

This document uses the common YANG types defined in [RFC6991].

This document does not allow to manage advanced PCP authentication features [RFC7652].

1.1. 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].

This document makes use of the terms defined in [RFC6887], [RFC7648], [RFC6970], and [RFC6970].

The terminology for describing YANG modules is defined in [RFC7950].

1.2. Tree Diagrams

The meaning of the symbols in these diagrams is as follows:

- 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), "ro" state data (read-only).

- Symbols after data node names: "?" means an optional node, "!" a container with presence, 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.

1.3. IP Address Format

Following the rationale defined in Section 5 of [RFC6887], this document uses IPv4-mapped IPv6 addresses to encode IPv4 addresses.

The all-zeros IPv6 address are expressed as (::).

The all-zeros IPv4 address is expressed by 80 bits of zeros, 16 bits of ones, and 32 bits of zeros (::ffff:0:0).

2. Overview of the PCP YANG Modules

The following sub-sections provide an overview of the PCP data models.

2.1. Common PCP

Common PCP YANG module groups a set of common definitions that are used in all PCP YANG modules.

2.2. PCP Client

Figure 1 depicts the YANG module for the PCP client.

```
module: ietf-pcp-client
    +--rw pcp-client
       +--rw enable?        boolean
       +--rw description?   string
       +--rw instances
          +--rw instance* [id]
             +--rw id                        uint32
             +--rw name?                     string
             +--rw capabilities
             |  +--rw supported-version* [version]
             |  |  +--rw version    uint8
             |  +--rw preferred-version?        uint8
```

```
|  +--rw authentication-support?   boolean
|  +--rw opcode-capability
|  |  +--rw map?        boolean
|  |  +--rw peer?       boolean
|  |  +--rw announce?   boolean
|  +--rw option-capability
|     +--rw third-party?      boolean
|     +--rw prefer-failure?   boolean
|     +--rw filter
|     |  +--rw filter-enabled?   boolean
|     |  +--rw max-filters?      uint32
|     +--rw port-set?         boolean
|     +--rw description
|     |  +--rw description-enabled?   boolean
|     |  +--rw max-description?       uint32
|     +--rw prefix64?         boolean
+--rw version* [version]
|  +--rw version    uint8
+--rw pcp-servers* [pcp-server-id]
|  +--rw pcp-server-id                uint32
|  +--rw pcp-server-ip-address* [address-id]
|  |  +--rw address-id    uint32
|  |  +--rw ip-address?   inet:ipv6-address
|  +--rw external-address-familly?    inet:ip-version
|  +--rw stale-external-ip-address?   inet:ipv6-prefix
+--rw authentication-enable?   boolean
+--rw opcode-configuration
|  +--rw map?        boolean
|  +--rw peer?       boolean
|  +--rw announce?   boolean
+--rw option-configuration
|  +--rw third-party?      boolean
|  +--rw prefer-failure?   boolean
|  +--rw filter
|  |  +--rw filter-enabled?   boolean
|  |  +--rw max-filters?      uint32
|  +--rw port-set?         boolean
|  +--rw description
|  |  +--rw description-enabled?   boolean
|  |  +--rw max-description?       uint32
|  +--rw prefix64?         boolean
+--rw mapping-table
|  +--rw mapping-entry* [index]
|     +--rw index                   uint32
|     +--rw status?                 enumeration
|     +--rw mapping-nonce?          string
|     +--rw internal-ip-address?    inet:ipv6-prefix
|     +--rw internal-port
```

```
|     |  +--rw start-port-number?   inet:port-number
|     |  +--rw end-port-number?     inet:port-number
|     +--rw external-ip-address?     inet:ipv6-prefix
|     +--rw external-port
|     |  +--rw start-port-number?   inet:port-number
|     |  +--rw end-port-number?     inet:port-number
|     +--rw protocol?                uint8
|     +--rw lifetime?                uint32
|     +--rw third-party-address?     inet:ipv6-prefix
|     +--rw filter* [filter-id]
|     |  +--rw filter-id             uint32
|     |  +--rw remote-ip-prefix?     inet:ipv6-prefix
|     |  +--rw remote-port-number?   inet:port-number
|     +--rw description?             string
|     +--rw prefer-failure-tagged?   boolean
+--rw traffic-statistics
   +--rw traffic-statistics
   |  +--rw sent-packet?      yang:zero-based-counter64
   |  +--rw sent-byte?        yang:zero-based-counter64
   |  +--rw rcvd-packet?      yang:zero-based-counter64
   |  +--rw rcvd-byte?        yang:zero-based-counter64
   |  +--rw dropped-packet?   yang:zero-based-counter64
   |  +--rw dropped-byte?     yang:zero-based-counter64
   +--rw opcode-statistics
   |  +--rw sent-map?         yang:zero-based-counter64
   |  +--rw rcvd-map?         yang:zero-based-counter64
   |  +--rw sent-peer?        yang:zero-based-counter64
   |  +--rw rcvd-peer?        yang:zero-based-counter64
   |  +--rw sent-annonce?     yang:zero-based-counter64
   |  +--rw rcvd-announce?    yang:zero-based-counter64
   |  +--rw rcvd-unknown?     yang:zero-based-counter64
   |  +--rw rcvd-malformed?   yang:zero-based-counter64
   +--rw mapping-table
      +--rw current-mt-size?   yang:zero-based-counter64
      +--rw max-mt-size?       uint32
```

Figure 1: PCP Client YANG Module

2.3. UPnP IGD/PCP Interworking Function

Figure 2 depicts the YANG module for the UPnP IGD-PCP IWF.

```
module: ietf-pcp-iwf
  augment /pcp-client:pcp-client/pcp-client:instances/pcp-client:instance/pcp-
client:capabilities:
    +--rw igd-supported-version* [igd-version]
       +--rw igd-version    enumeration
  augment /pcp-client:pcp-client/pcp-client:instances/pcp-client:instance:
    +--rw igd-version
       +--rw igd-version?   enumeration
  augment /pcp-client:pcp-client/pcp-client:instances/pcp-client:instance/pcp-
client:mapping-table/pcp-client:mapping-entry:
    +--rw igd-control-point-address?   inet:ip-address
    +--rw igd-control-point-port?      inet:port-number
```

Figure 2: IWF YANG Module

2.4. PCP Proxy

Figure 3 depicts the YANG module for the PCP proxy.

```
module: ietf-pcp-proxy
  augment /pcp-client:pcp-client/pcp-client:instances/pcp-client:instance/pcp-
client:option-configuration:
    +--rw relay-mandatory-unknown-option?   boolean
    +--rw relay-optionnal-unknown-option?   boolean
  augment /pcp-client:pcp-client/pcp-client:instances/pcp-client:instance:
    +--rw terminate-proxy-recursion?   boolean
  augment /pcp-client:pcp-client/pcp-client:instances/pcp-client:instance/pcp-
client:mapping-table/pcp-client:mapping-entry:
    +--rw local-assigned-ip-address?   inet:ipv6-prefix
    +--rw local-assigned-port
       +--rw start-port-number?   inet:port-number
       +--rw end-port-number?     inet:port-number
```

Figure 3: PCP Proxy YANG Module

2.5. PCP Server

Figure 4 depicts the YANG module for the PCP server.

```
module: ietf-pcp-server
    +--rw pcp-serve
       +--rw enable?      boolean
       +--rw instances
          +--rw instance* [id]
             +--rw id                                uint32
             +--rw name?                             string
             +--rw capabilities
```

```
|  +--rw supported-version* [version]
|  |  +--rw version    uint8
|  +--rw preferred-version?                    uint8
```

```
|  +--rw authentication-support?              boolean
|  +--rw opcode-capability
|  |  +--rw map?        boolean
|  |  +--rw peer?       boolean
|  |  +--rw announce?   boolean
|  +--rw option-capability
|  |  +--rw third-party?      boolean
|  |  +--rw prefer-failure?   boolean
|  |  +--rw filter
|  |  |  +--rw filter-enabled?   boolean
|  |  |  +--rw max-filters?      uint32
|  |  +--rw port-set?         boolean
|  |  +--rw description
|  |  |  +--rw description-enabled?   boolean
|  |  |  +--rw max-description?       uint32
|  |  +--rw prefix64?         boolean
|  +--rw port-randomization-support?          boolean
|  +--rw port-preservation-suport?            boolean
|  +--rw port-parity-preservation-support?    boolean
|  +--rw protocol-capabilities* [protocol-id]
|  |  +--rw protocol-id    uint8
|  +--rw pcp-controlled-function-capability
|     +--rw pcp-controlled-function*   identityref
+--rw version* [version]
|  +--rw version    uint8
+--rw pcp-server-ip-address* [address-id]
|  +--rw address-id    uint32
|  +--rw ip-address?   inet:ipv6-address
+--rw authentication-enable?            boolean
+--rw opcode-configuration
|  +--rw map?        boolean
|  +--rw peer?       boolean
|  +--rw announce?   boolean
+--rw option-configuration
|  +--rw third-party?       boolean
|  +--rw prefer-failure?    boolean
|  +--rw filter
|  |  +--rw filter-enabled?   boolean
|  |  +--rw max-filters?      uint32
|  +--rw port-set-option
|  |  +--rw port-set-enable?         boolean
|  |  +--rw default-port-set-size?   uint16
|  |  +--rw maximum-port-set-size?   uint16
|  +--rw description
|  |  +--rw description-enabled?   boolean
|  |  +--rw max-description?       uint32
|  +--rw prefix64-option
|     +--rw prefix64-option-enable?   boolean
```

```
|     +--rw prefix64* [prefix64-id]
|        +--rw prefix64-id         uint32
|        +--rw prefix64?           inet:ipv6-prefix
|        +--rw suffix?             yang:hex-string
|        +--rw dest-ipv4-prefix* [ipv4-prefix-id]
|           +--rw ipv4-prefix-id    uint32
|           +--rw ipv4-prefix?      inet:ipv4-prefix
+--rw port-randomization-enable?          boolean
+--rw port-preservation-enable?           boolean
+--rw port-parity-preservation-enable?    boolean
+--rw nonce-validation-checks-enable?     boolean
+--rw subscriber-mask?                    uint8
+--rw port-quota?                         uint16
+--rw exclude-ports* [id]
|  +--rw id                    uint16
|  +--rw start-port-number?    inet:port-number
|  +--rw end-port-number?      inet:port-number
+--rw protocol* [protocol-id]
|  +--rw protocol-id    uint8
+--rw epoch-set?                         uint32
+--rw lifetime
|  +--rw minimum-lifetime?   uint32
|  +--rw maximum-lifetime?   uint32
+--rw error-lifetime
|  +--rw minimum-error-lifetime?   uint32
|  +--rw maximum-error-lifetime?   uint32
+--rw mapping-table
|  +--rw mapping-entry* [index]
|     +--rw index                   uint32
|     +--rw status?                 enumeration
|     +--rw mapping-nonce?          string
|     +--rw internal-ip-address?    inet:ipv6-prefix
|     +--rw internal-port
|     |  +--rw start-port-number?   inet:port-number
|     |  +--rw end-port-number?     inet:port-number
|     +--rw external-ip-address?    inet:ipv6-prefix
|     +--rw external-port
|     |  +--rw start-port-number?   inet:port-number
|     |  +--rw end-port-number?     inet:port-number
|     +--rw protocol?               uint8
|     +--rw lifetime?               uint32
|     +--rw third-party-address?    inet:ipv6-prefix
|     +--rw filter* [filter-id]
|     |  +--rw filter-id            uint32
|     |  +--rw remote-ip-prefix?    inet:ipv6-prefix
|     |  +--rw remote-port-number?  inet:port-number
|     +--rw description?            string
|     +--rw prefer-failure-tagged?  boolean
```

```
               +--rw traffic-statistics
                  +--rw traffic-statistics
                  |  +--rw sent-packet?      yang:zero-based-counter64
                  |  +--rw sent-byte?        yang:zero-based-counter64
                  |  +--rw rcvd-packet?      yang:zero-based-counter64
                  |  +--rw rcvd-byte?        yang:zero-based-counter64
                  |  +--rw dropped-packet?   yang:zero-based-counter64
                  |  +--rw dropped-byte?     yang:zero-based-counter64
                  +--rw opcode-statistics
                  |  +--rw sent-map?         yang:zero-based-counter64
                  |  +--rw rcvd-map?         yang:zero-based-counter64
                  |  +--rw sent-peer?        yang:zero-based-counter64
                  |  +--rw rcvd-peer?        yang:zero-based-counter64
                  |  +--rw sent-annonce?     yang:zero-based-counter64
                  |  +--rw rcvd-announce?    yang:zero-based-counter64
                  |  +--rw rcvd-unknown?     yang:zero-based-counter64
                  |  +--rw rcvd-malformed?   yang:zero-based-counter64
                  +--rw mapping-table
                  |  +--rw current-mt-size?   yang:zero-based-counter64
                  |  +--rw max-mt-size?       uint32
                  +--rw port-in-use?          percent
```

Figure 4: PCP Server YANG Module

3. YANG Modules

3.1. Common PCP Module

```
<CODE BEGINS> file "ietf-pcp@2017-10-17.yang"
module ietf-pcp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-pcp";
  prefix pcp;

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

  organization "xxx Working Group";
  contact
    "Mohamed Boucadair <mohamed.boucadair@orange.com>
     Christian Jacquenet <christian.jacquenet@orange.com>";

  description
    "This module embeds the core PCP characteristics, including
     the description of PCP operations, options and mapping entries.

     Copyright (c) 2017 IETF Trust and the persons identified as
```

```
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC XXXX; see
      the RFC itself for full legal notices.";

    revision 2017-10-17 {
      description "Align with NMDA";
      reference "-05";
    }

    revision 2015-08-05 {
      description "Changes tbc.";
      reference "-00";
    }

  /*
   * Identities
   */

    identity c_function {
      description
        "Base identity for controlled function.";

      reference
        "RFC 3022.";
    }


    identity nat44 {
      base pcp:c_function;
      description
        "Base identity for NAT44 type.";

      reference
        "RFC 3022.";
    }

    identity nat64 {
      base pcp:c_function;
      description
        "Base identity for NAT64 type.";
```

```
      reference
        "RFC 6146.";
    }

    identity dslite {
      base pcp:c_function;
      description
        "Base identity for DS-Lite type.";

      reference
        "RFC 6333.";
    }

    identity nptv6 {
      base pcp:c_function;
      description
        "Base identity for NPTv6 type.";

      reference
        "RFC 6296.";
    }

    identity ipv4-firewall {
      base pcp:c_function;
      description
        "Base identity for IPv4 firewall type.";
    }

    identity ipv6-firewall {
      base pcp:c_function;
      description
        "Base identity for IPv6 firewall type.";
    }

    identity port-range-router {
      base pcp:c_function;
      description
        "Base identity for Port Range Router type.";

      reference
        "RFC 6346.";
    }


  /*
   * Grouping
   */
```

```
   //Description option

   grouping description-option {
     description
       "used to configure DESCRIPTION option.";

     leaf description-enabled {
       type boolean;

       description
         "Enable/disable DESCRIPTION option.";

       reference
         "RFC 7220";
     }

     leaf max-description {
       type uint32;

       description
         "Indicates the maximum length of the description
          associated with a mapping.";

       reference
         "RFC 7220";
     }
   }

   //Filter option

   grouping filter-option {
     description
       "FILTER option";

     leaf filter-enabled {
       type boolean;

       description
         "Enable/disable FILTER option.";

       reference
         "RFC 6887";
     }

     leaf max-filters {
       type uint32;

       description
```

```
        "Indicates the maximum number of filters
         associated with a mapping.";

      reference
        "RFC 6887";
    }
  }

  // Port set option

  grouping port-set-option {
    description
      "PORT_SET option.";

    leaf port-set-enable {
      type boolean;

      description
        "Enable/disable PORT_SET option.";

      reference
        "RFC 7753";
    }

    leaf default-port-set-size {
      type uint16;

      description
        "Indicates the default size of a port set.";

      reference
        "RFC 7753";
    }

    leaf maximum-port-set-size {
      type uint16;

      description
        "Indicates the maximum size of a port set.";

      reference
        "RFC 7753";
    }
  }


  //Opcodes
```

```
   grouping opcode {
     description
       "Indicates the set of supported/enabled PCP opcodes.";

     leaf map {
       type boolean;

       description
         "MAP opcode";

       reference
         "RFC 6887";
     }

     leaf peer {
       type boolean;

       description
         "PEER opcode";

       reference
         "RFC 6887";
     }

     leaf announce {
       type boolean;

       description
         "ANNOUNCE opcode.";

       reference
         "RFC 6887";
     }
   }

   //Options

   grouping option {
     description
       "a set of PCP options.";

     leaf third-party {
       type boolean;

       description
         "THIRD_PARTY option is used when a PCP client wants
          to control a mapping to an internal host other
          than itself.";
```

```
      reference
        "RFC 6887";

    }

    leaf prefer-failure {
      type boolean;

      description
        "This option indicates that if the PCP server is unable
         to map both the suggested external port and suggested
         external address, the PCP server should not create
         a mapping.  This differs from the behavior without this
         option, which is to create a mapping.

         PREFER_FAILURE is never necessary for a PCP client to
         manage mappings for itself, and its use causes
         additional work in the PCP client and in the PCP
         server. See Section 13.2 of [RFC6887].";

      reference
        "Section 13.2 ofRFC 6887";
    }

    container filter {
      description
        "This option indicates that filtering incoming packets
         is desired.";

         uses filter-option;
    }

    leaf port-set {
      type boolean;

      description
        "Indicates whether PORT_SET is supported/enabled.";
    }

    container description {
      description
        "Associates a description with a mapping.";

      uses description-option;

      reference
        "RFC 7220";
    }
```

```
    leaf prefix64 {
      type boolean;

      description
        "PREFIX64 PCP option.";

      reference
        "RFC 7225";
    }
  }

  // port numbers: single or port range

  grouping port-number {
    description
      "Individual port or a range of ports.
       When only start-port-numbert is present,
       it represents a single port.";

    leaf start-port-number {
      type inet:port-number;

      description
        "Begining of the port range.";

      reference
        "Section 3.2.9 of RFC 8045.";
      }

    leaf end-port-number {
      type inet:port-number;

      must ". >= ../start-port-number"
        {
          error-message
             "The end-port-number must be greater than or
              equal to start-port-number.";
         }

       description
         "End of the port range.";

      reference
          "Section 3.2.10 of RFC 8045.";
    }
  }

  // Filter
```

```
   grouping filter {
     description
       "The remote peer IP address and remote peer port of
        the FILTER option indicate the permitted remote peer's
        source IP address and source port for packets from
        the Internet; other traffic from other addresses
        is blocked.";

     leaf filter-id {
       type uint32;

       description
         "An identifier of the filter.";
     }

     leaf remote-ip-prefix {
       type inet:ipv6-prefix;

       description
         "The IP address of the remote peer.";
       }

     leaf remote-port-number {
       type inet:port-number;

       description
         "The port number of the remote peer. Value 0
          indicates 'all ports'.";
     }
   }

   // PCP mapping entry

   grouping mapping-entry {
     description
       "A PCP mapping entry.";

    leaf index {
       type uint32;

       description
         "A unique identifier of a mapping entry.";
      }

     leaf status {
       type enumeration {

         enum "disabled"  {
```

```
          description
            "The mapping entry is not in use (Disabled).";
        }

        enum "requested" {
          description
            "A PCP request has been sent for this mapping.
             Still waiting for a response from the server.";
        }

        enum "assigned" {
          description
            "This mapping has been granted by the server.";
        }

        enum "stale" {
          description
            "This is a stale mapping (case of reboot).";
        }
      }
      description
        "Indicates the status of a mapping entry.";
    }

    leaf mapping-nonce {
      type string;

      description
        "A random value chosen by the PCP client";
    }

    leaf internal-ip-address {
      type inet:ipv6-prefix;

      description
        "Corresponds to the PCP Client's IP Address
         defined in [RFC6887].";
    }

    container internal-port {
      description
        "Internal port for the mapping. Value 0 indicates
         'all ports', and is legal when the lifetime is zero
         (a delete request), if the protocol does not use
         16-bit port numbers, or the client is requesting
         'all ports'.  If the protocol is zero
         (meaning 'all protocols'), then internal port
         is set to zero.";
```

```
        uses port-number;
      }

      leaf external-ip-address {
        type inet:ipv6-prefix;

        description
          "External IP address. Can be 'Suggested' or 'Assigned'.

           It can be set by a client to stale-ip-address, if available
           or to (::) (for requesting external IPv6 addresses)
           or (::ffff:0:0) (for requesting external IPv4 addresses).";
      }

      container external-port {
        description
          "External port number. Can be 'Suggested' or 'Assigned'.";

        uses port-number;
      }

      leaf protocol {
        type uint8;

        description
          "Upper-layer protocol associated with this Opcode.
           Values are taken from the IANA protocol registry.
           For example, this field contains 6 (TCP) if the Opcode
           is intended to create a TCP mapping.  This field contains
           17 (UDP) if the Opcode is intended to create a UDP mapping.

           The value 0 has a special meaning for 'all protocols'.";
      }

      leaf lifetime {
        type uint32;

        description
          "Lifetime of the mapping.

           Can be requested/assigned/remaining";
      }

      leaf third-party-address {
         type inet:ipv6-prefix;

         description
           "used to indicate the internal IP address
```

```
           when THIRD_PARTY is in use.";
      }

      list filter {
        key filter-id;

        description
          "a list of filters associated with the mapping.";

         uses filter;
       }

      leaf description {
        type string;

        description
          "a description string associated with the mapping.";
      }

      leaf prefer-failure-tagged {
        type boolean;

        description
          "a tag which indicates whether PREFER_FAILURE
           is (to be) used.";
      }
    }

    // PCP result code

    grouping status-code {
      description
        "stores the result status code";

      leaf status-code {
         type enumeration {
           enum "SUCCESS" {
             description
               "Success";
           }

           enum "unsupported-version" {
             description
               "The version number at the start of the PCP Request
                header is not recognized by this PCP server.
                This is a long lifetime error.";
           }
```

```
          enum "not-authorized" {
            description
              "The requested operation is disabled for this PCP
               client, or the PCP client requested an operation
               that cannot be fulfilled by the PCP server's
               security policy.

               This is a long lifetime error.";
          }

          enum "malformed-request" {
            description
              "The request could not be successfully parsed.

               This is a long lifetime error.";
        }

        enum "unsupported-opcode" {
          description
            "Unsupported Opcode.
             This is a long lifetime error.";
        }

        enum "unsupported-option" {
          description
            "Unsupported option.  This error only occurs if
             the option is in the mandatory-to-process range.

             This is a long lifetime error.";
        }

        enum "malformed-option" {
          description
            "Malformed option (e.g., appears too many times,
             invalid length).

             This is a long lifetime error.";
        }

        enum "network-failure" {
          description
            "The PCP server or the device it controls is
             experiencing a network failure of some sort
             (e.g., has not yet obtained an external
             IP address).

             This is a short lifetime error.";
        }
```

```
          enum "no-resources" {
            description
              "Request is well-formed and valid, but the server
               has insufficient resources to complete
               the requested operation at this time.

               For example, the NAT device cannot create more
               mappings at this time, is short of CPU cycles
               or memory, or is unable to handle the request
               due to some other temporary condition.

               The same request may succeed in the future.
               This is a system-wide error, different from
               USER_EX_QUOTA.  This can be used as a
               catch-all error, should no other error
               message be suitable.

               This is a short lifetime error.";
          }

          enum "unsupported-protocol" {
            description
              "Unsupported transport protocol, e.g.,
               SCTP in a  NAT that handles only UDP and TCP.

               This is a long lifetime error.";
          }

            enum "ex-quota" {
                description
                   "This attempt to create a new mapping would
                    exceed this subscriber's port quota.

                    This is a short lifetime error.";
            }

            enum "cannot-provide-external" {
                 description
                    "The suggested external port and/or
                     external address cannot be provided.
                     This error must only be returned for:
                      *  MAP requests that included the
                         PREFER_FAILURE option
                      *  MAP requests for the SCTP protocol
                        (PREFER_FAILURE is implied)
                      *  PEER requests.";
            }
```

```
              enum "address-mismatch" {
                  description
                    "The source IP address of the request
                     packet does not match the contents of the
                     PCP Client's IP Address field, due to an
                     unexpected NAT on the path between the PCP
                     client and the PCP-controlled NAT or firewall.

                     This is a long lifetime error.";
              }

              enum "extensive-remote-peer" {
                  description
                     "The PCP server was not able to create the
                      filters in this request.  This result code must
                      only be returned if the MAP request contained
                      the FILTER option.

                      This is a long lifetime error.";
              }
          }
          description
              "result status code.";
       }
   }

   // PCP servers list

   grouping pcp-server-address {

       description
             "A list of PCP servers. Each PCP server can be identified
              by one or multiple IP addresses.";

       leaf pcp-server-id {
         type uint32;
         description
              "A unique identifier.";
       }

       list pcp-server-ip-address {

           key address-id;

           description
               "a list of IP addresses of a PCP server";

           leaf address-id {
```

```
                type uint32;
                description
                  "An identifier";
           }

           leaf ip-address {
               type inet:ipv6-address;
               description
                 "An IP address of a PCP server.";
           }
       }

       leaf external-address-familly {
           type inet:ip-version;
               description
                   "The address family of the external address(es)
                    managed by the PCP server.
                    Can be IPv4, IPv6 or both.";
      }

       leaf stale-external-ip-address {
          type inet:ipv6-prefix;
              description
                 "A stale address that can be used by the PCP client
                  to be assigned the same address upon reboot
                  or other failure events.";
       }
   }

   // status of the communication with configured PCP servers

   grouping pcp-server-address-status {

     description
           "Groups the status of the communication between
            a PCP client a server.";

     uses pcp-server-address;

     leaf source {
           type enumeration {
               enum "manual-configuration"{
               description
                 "The server has been manually configured.";
               }

               enum "dhcpv6"{
               description
```

```
                "Retrieved from DHCPv6 [RFC7291].";
              }

              enum "dhcpv4"{
              description
                 "Retrieved from DHCPv4 [RFC7291].";
              }

              enum "else"{
              description
                "Else (e.g., TR-96.)";
              }
            }
       description
         "source of the PCP server reachability information.";
       }

     leaf in-use {
        type boolean;
        description
          "Indicates whether this in-use instance of the server
           is the result of the selection
           process defined in [RFC7488].";
     }

     leaf server-epoch {
        type uint32;
        description
           "The PCP server's Epoch.";
     }

     leaf client-epoch {
        type uint32;
        description
           "The PCP client's Epoch.";
     }

     leaf current-version {
        type uint8;
        description
          "The version that is selected as per the version negotiation
           procedure specified in Section 9 of [RFC6877].";
     }
   }

   // type of the PCP-controlled function.

   grouping pcp-controlled-function {
```

```
       description
          "A set of PCP-controlled functions.
           One or multiple functions can be controlled
           by the same PCP server. ";

           leaf-list pcp-controlled-function {
             type identityref {
               base c_function;
             }
             description
               "Type of NAT.";
           }
   }


   // traffic statistics

   grouping traffic-stat {
      description
            "Groups a set of statistics.";

       container traffic-statistics {
           description
             "Generic traffic statistics.";

           leaf sent-packet {
             type yang:zero-based-counter64;
              description
               "Packets sent";
           }

           leaf sent-byte {
             type yang:zero-based-counter64;
             description
               "Counter for sent traffic in bytes.";
           }

           leaf rcvd-packet {
             type yang:zero-based-counter64;
             description
                "Counter for received packets.";
           }

           leaf rcvd-byte {
             type yang:zero-based-counter64;
             description
               "Counter for received traffic in bytes.";
           }
```

```
          leaf dropped-packet {
            type yang:zero-based-counter64;
            description
              "Counter for dropped packets.";
          }

          leaf dropped-byte {
            type yang:zero-based-counter64;
            description
              "Counter for dropped traffic in bytes.";
          }
        }

    container opcode-statistics {
          description
                "Opcode-related statistics.";

          leaf sent-map {
              type yang:zero-based-counter64;
              description
                "Counter for sent MAP messages";
          }

          leaf rcvd-map {
              type yang:zero-based-counter64;
              description
                "Counter for received MAP messages";
          }

          leaf sent-peer {
              type yang:zero-based-counter64;
              description
                "Counter for sent PEER messages";
          }

          leaf rcvd-peer {
              type yang:zero-based-counter64;
              description
                "Counter for received PEER messages";
          }

          leaf sent-annonce {
              type yang:zero-based-counter64;
              description
                "Counter for sent ANNOUNCE messages";
          }

          leaf rcvd-announce {
```

```
              type yang:zero-based-counter64;
              description
                "Counter for received ANNOUNCED messages";
            }

            leaf rcvd-unknown {
              type yang:zero-based-counter64;
                description
                  "Counter for received unknown opcodes";
            }

            leaf rcvd-malformed {
              type yang:zero-based-counter64;
                description
                   "Counter for received malformed opcodes";
            }
        }
    }

    // mapping table statistics

    grouping mapping-table-stats {
        description
            "PCP mapping table related statistics.";

        leaf current-mt-size {
          type yang:zero-based-counter64;
          description
              "Size of the mapping table";
        }

        leaf max-mt-size {
          type uint32;
          description
              "Maximum configured size of the mapping table.";
        }
    }


    // PCP versions

    grouping pcp-version {
        description
         "PCP version(s)";

        leaf version {
          type uint8;
          description
```

```
            "Indicates a PCP server.
              Current versions are: 0, 1, and 2.";
        }
    }
    }

    <CODE ENDS>
```

3.2. PCP Client

```
    <CODE BEGINS> file "ietf-pcp-client@2017-10-17.yang"
    module ietf-pcp-client {
      yang-version 1.1;

      namespace "urn:ietf:params:xml:ns:yang:ietf-pcp-client";
      prefix pcp-client;

      import ietf-pcp { prefix pcp; }

      organization "N/A Working Group";
      contact
        "Mohamed Boucadair <mohamed.boucadair@orange.com>
         Christian Jacquenet <christian.jacquenet@orange.com>";

      description
        "This module contains a collection of YANG definitions for
         PCP client implementations.

         Copyright (c) 2017 IETF Trust and the persons identified as
         authors of the code.  All rights reserved.

         Redistribution and use in source and binary forms, with or
         without modification, is permitted pursuant to, and subject
         to the license terms contained in, the Simplified BSD License
         set forth in Section 4.c of the IETF Trust's Legal Provisions
         Relating to IETF Documents
         (http://trustee.ietf.org/license-info).

         This version of this YANG module is part of RFC XXXX; see
         the RFC itself for full legal notices.";

      revision 2017-10-17 {
        description "Align with NMDA";
        reference "-05";
      }

      revision 2015-08-05 {
          description "Changes tbc.";
```

```
          reference "tbc";
      }

    /*
     *PCP Client
     */

     container pcp-client {
       description
         "PCP client ";

        leaf enable {
          type boolean;

           description
             "Enable/disable the PCP client.";
        }

        leaf description {
          type string;

          description
            "Associated a description with the module.";
        }

        container instances {
          description
            "A set of PCP client instances.";

          list instance {
            key "id";

            description
              "A PCP client instance.";

            leaf id {
              type uint32;

              description
                "An identifier of the PCP client instance.";
            }

            leaf name {
              type string;

              description
                "A name of the PCP client instance.";
            }
```

```
            container capabilities {
              description "Capabilities";

              list supported-version {
                key version;

                description
                  "list of supported PCP versions";

                uses pcp:pcp-version;
              }

              leaf preferred-version {
                type uint8;

                description
                  "The preferred version configured
                   by an administrator.";
              }

              leaf authentication-support {
                type boolean;

                description
                  "Indicates whether PCP authentication is
                   supported.";
              }

              container opcode-capability {
                description
                  "Opcode-related capabilities.";

                uses pcp:opcode;
              }

              container option-capability {
                description
                  "Option-related capabilities";

                uses pcp:option;
              }
            }

            list version {
              key version;

              description
                "Indicates the set of supported PCP versions
```

```
          (0, 1, 2)";

        uses pcp:pcp-version;
      }

      list pcp-servers {
        key "pcp-server-id";

        description
          "List of provisioned PCP servers.";

        uses pcp:pcp-server-address;
      }

      leaf authentication-enable {
        type boolean;

        description
          "Enable/Disable PCP authentication.";
      }

      container opcode-configuration {
        description
          "Opcode-related configuration";

        uses pcp:opcode;
      }

      container option-configuration {
        description
          "Options-related configuration.";

        uses pcp:option;
      }

      container mapping-table {
        description
          "Mapping table maintained by a PCP client
           instance.";

        list mapping-entry {
          key "index";

          description
            "PCP Mapping entry.";

          uses pcp:mapping-entry;
        }
```

```
            }

            container traffic-statistics {
              description
                "traffic statistics.";

              uses pcp:traffic-stat;

              container mapping-table {
                description
                  "mapping table related statistics.";

                uses pcp:mapping-table-stats;
              }
            }
          }
        }
      }
    }
```

<CODE ENDS>

3.3. UPnP IGD/PCP Interworking Function

```
<CODE BEGINS> file "ietf-pcp-iwf@2017-10-17.yang"
module ietf-pcp-iwf {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-pcp-iwf";
  prefix pcp-iwf;

  import ietf-inet-types { prefix inet; }
  import ietf-pcp-client { prefix pcp-client; }

  organization "xxxx Working Group";
  contact
     "Mohamed Boucadair <mohamed.boucadair@orange.com>
      Christian Jacquenet <christian.jacquenet@orange.com>";

  description
       "This module contains a collection of YANG definitions for
        UPnP IGD/PCP Interworking implementations.

        Copyright (c) 2017 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
```

```
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

  revision 2017-10-17 {
    description "Align with NMDA";
    reference "-05";
  }

  revision 2015-08-05 {
      description "Changes xxxx.";
       reference "xxxx";
   }


// IGD versions

grouping igd-version {
    description
      "UPnp IGD Version";

   leaf igd-version {

      type enumeration {

         enum "igd:1" {
             description
               "UPnP IGD:1";
         }

         enum "igd:2" {
             description
              "UPnP IGD:2";
         }

         enum "both" {
             description
              "UPnP IGD:1 and UPnP IGD:2";
         }
      }
   description
    "UPnP IGD Version";
 }
```

```
}


augment "/pcp-client:pcp-client/pcp-client:instances/pcp-client:instance/pcp-
client:capabilities" {
           description "Capabilities";

          list igd-supported-version {

            key igd-version;

             description
               "list of supported IGD versions";

             uses igd-version;
           }

}

augment "/pcp-client:pcp-client/pcp-client:instances/pcp-client:instance" {
  description
    "IGD version(s)";

   container igd-version {
     description
       "Configure UPnP IGD version(s).";

     uses igd-version;
   }
}

augment "/pcp-client:pcp-client/pcp-client:instances/pcp-client:instance/pcp-
client:mapping-table/pcp-client:mapping-entry" {
  description
    "Mapping table as maintained by a
     UPnP IGD/PCP IWF instance";

  leaf igd-control-point-address {
    type inet:ip-address;

    description
      "IP address of the UPnP Control Point.";
  }

  leaf igd-control-point-port {
    type inet:port-number;

    description
      "Port number of the UPnP Control Point.";
```

```
  }
```

```
 }
}

<CODE ENDS>
```

3.4. PCP Proxy

```
<CODE BEGINS> file "ietf-pcp-proxy@2017-10-17.yang"
module ietf-pcp-proxy {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-pcp-proxy";
  prefix pcp-proxy;

  import ietf-inet-types { prefix inet; }
  import ietf-pcp { prefix pcp; }
  import ietf-pcp-client { prefix pcp-client; }

  organization "xxxx Working Group";
  contact
     "Mohamed Boucadair <mohamed.boucadair@orange.com>
      Christian Jacquenet <christian.jacquenet@orange.com>";

  description
       "This module contains a collection of YANG definitions for
        PCP Proxy implementations.

        Copyright (c) 2017 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

  revision 2017-10-17 {
    description "Align with NMDA";
    reference "-05";
  }

  revision 2015-08-05 {
      description "Changes xxxx.";
       reference "xxxx";
```

```
   }

augment "/pcp-client:pcp-client/pcp-client:instances/pcp-client:instance/pcp-
client:option-configuration" {
  description
    "Augment the PCP client module with proxy
     specific parameters: instruct the behavior
     with regards to unknown options.";


  leaf relay-mandatory-unknown-option {
    type boolean;

    description
      "The proxy can be instructed to relay or
       to reject mandatory unknown options.";
  }

  leaf relay-optionnal-unknown-option {
    type boolean;

    description
      "The proxy can be instructed to relay or
       to reject optional unknown options.";
  }
}

augment "/pcp-client:pcp-client/pcp-client:instances/pcp-client:instance" {
  description
    "Instruct the proxy to terminate recusion.";

  leaf terminate-proxy-recursion {
    type boolean;

    description
      "The proxy can be instructed to terminate
       proxy recursion.";
  }
}


augment "/pcp-client:pcp-client/pcp-client:instances/pcp-client:instance/pcp-
client:mapping-table/pcp-client:mapping-entry" {
  description
    "Augment the local mapping table with locally
     assigned parameters.";


  leaf local-assigned-ip-address {
```

```
    type inet:ipv6-prefix;
```

```
    description
      "If the local PCP-controlled function
       alters the source IP address, this
       information must be stored.";
  }

  container local-assigned-port {
     description
        "If the local PCP-controlled function
         alters the source port, this
         information must be stored.";

     uses pcp:port-number;
  }
 }
}
```

<CODE ENDS>

3.5. PCP Server

```
   <CODE BEGINS> file "ietf-pcp-server@2017-10-17.yang"
   module ietf-pcp-server {
      yang-version 1.1;

       namespace "urn:ietf:params:xml:ns:yang:ietf-pcp-server";
       prefix pcp-server;

       import ietf-inet-types { prefix inet; }
       import ietf-yang-types { prefix yang; }
       import ietf-pcp { prefix pcp; }

       organization "xxxx Working Group";
       contact
        "Mohamed Boucadair <mohamed.boucadair@orange.com>
         Christian Jacquenet <christian.jacquenet@orange.com>";

      description
         "This module contains a collection of YANG definitions for
           PCP server implementations.

           Copyright (c) 2017 IETF Trust and the persons identified as
           authors of the code.  All rights reserved.

           Redistribution and use in source and binary forms, with or
           without modification, is permitted pursuant to, and subject
```

```
          to the license terms contained in, the Simplified BSD License
          set forth in Section 4.c of the IETF Trust's Legal Provisions
          Relating to IETF Documents
          (http://trustee.ietf.org/license-info).

          This version of this YANG module is part of RFC XXXX; see
          the RFC itself for full legal notices.";

    revision 2017-10-17 {
      description "Align with NMDA";
      reference "-05";
    }

      revision 2015-08-05 {
        description "Changes xxxx.";
         reference "xxxx";
      }

  // Typedef

      typedef percent {
          type uint8 {
              range "0 .. 100";
          }
          description
           "Percentage";
      }

  /*
   * Grouping
   */

  // Port set option

  grouping port-set-option {
      description
                  "PORT_SET option.";

      leaf port-set-enable {
         type boolean;
         description
            "Enable/disable PORT_SET option.";
      }

      leaf default-port-set-size {
         type uint16;
         description
            "Indicates the default size of a port set.";
```

```
      }

      leaf maximum-port-set-size {
         type uint16;
         description
            "Indicates the maximum size of a port set.";
      }
   }

   // Prefix64 port set

   grouping prefix64-option {
       description
              "PREFIX64 option as defined in [RFC7225].";

       leaf prefix64-option-enable {
           type boolean;
           description
              "Indicates whether the option is enabled/disabled.";
       }

       list prefix64 {
           key "prefix64-id";
           description
                    "maintains a list of Prefix64s.";

           leaf prefix64-id {
                 type uint32;
                 description
                    "An identifier of a Prefix64.";
           }

           leaf prefix64 {
                type inet:ipv6-prefix;
                description
                    "A Prefix64";
           }

           leaf suffix {
                type yang:hex-string;
                description
                  "The suffix is used for constructing an
                    IPv4-converted IPv6 address from an IPv4 address as
                    specified in Section 2.2 of [RFC6052]. No suffix is
                    included if a /96 Prefix64 is used.";
           }

           list dest-ipv4-prefix {
```

```
                key "ipv4-prefix-id";
                description
                   "used to solve the destination-dependent
                     Pref64::/n discovery problem discussed in
                     Section 5.1 of [RFC7050].";

                leaf ipv4-prefix-id {
                     type uint32;
                    description
                      "An identifier of a destination IPv4 prefix";
                }

                leaf ipv4-prefix {
                     type inet:ipv4-prefix;
                     description
                    "an IPv4 prefix.";
                }
            }
        }
    }

    //option list: server side

    grouping option-server {
       description
         "Used for option-related operations
           at the server's side.";

       leaf third-party {
           type boolean;
           description
             "enable/disable THIRD_PARTY option.";
       }

       leaf prefer-failure {
           type boolean;
           description
              "enable/disable PREFER_FAILURE option.";
       }

       container filter {
           description
            "enable/disable FILTER option.";

           uses pcp:filter-option;
       }

       container port-set-option {
```

```
          description
            "enable/disable PORT_SET option.";

          uses pcp:port-set-option;
       }

       container description {
          description
            "enable/disable DESCRIPTION option.";
          uses pcp:description-option;
       }

       container prefix64-option {
          description
            "enable/disable PREFIX64 option.";
          uses prefix64-option;
       }
    }

    /*
      * PCP Server Instances
      */

    container pcp-serve {
        description
         "PCP server";

        leaf enable {
            type boolean;
            description
             "Enable/Disable PCP server function.";
        }

        container instances {
           description
             "PCP server instances";

            list instance {
                key "id";
                  description
                    "a PCP server instance.";

              leaf id {
                 type uint32;
                 description
                   "PCP server instance identifier.";
              }
```

```
              leaf name {
                 type string;
                 description
                   "A name associated with the PCP server instance";
              }

    container capabilities {
       description
          "Capabilities";

       list supported-version {
                 key version;
                 description
                    "List of supported PCP versions.";

                 uses pcp:pcp-version;
              }

       leaf preferred-version {
                 type uint8;
                 description
                   "List of preferred version.
                     Mainly used for unsolicited messages.";
              }


         leaf authentication-support {
           type boolean;
           description
            "Status of the support of PCP authentication";
         }

         container opcode-capability {
           description
            "Opcode-related capabilities";
           uses pcp:opcode;
         }

         container option-capability {
           description
            "Option-related capabilities";

           uses pcp:option;
         }

          leaf port-randomization-support {
             type boolean;
             description
```

```
            "Indicates whether port randomization is
              supported.";
          }

          leaf port-preservation-suport {
             type boolean;
             description
              "Indicates whether port preservation
                is supported.";
          }

          leaf port-parity-preservation-support {
             type boolean;
             description
              "Indicates whether port parity preservation is
                supported.";
          }

          list protocol-capabilities {
            key "protocol-id";
            description
              "A set of supported transported protocols";

            leaf protocol-id {
               type uint8;
               description
                 "transport protocol";
            }
          }

          container pcp-controlled-function-capability {
            description
              "list of controlled functions.";

            uses pcp:pcp-controlled-function;
        }
      }


            list version {
               key version;
               description
                 "Indicates the PCP version(s) supported by the
                   PCP server.
                   Current supported versions are 0, 1, and 2.";

               uses pcp:pcp-version;
            }
```

```
            list pcp-server-ip-address {

                key address-id;

                description
                 "set one or multiple IP addresses for
                  the PCP server";

                leaf address-id {
                  type uint32;
                  description
                     "The identifier of the address";
                }

                leaf ip-address {
                  type inet:ipv6-address;
                  description
                    "IP (v4/v6) address of the PCP server";
               }
             }

            leaf authentication-enable {
              type boolean;
              description
               "Enable/disable PCP authentication";
            }

            container opcode-configuration {
                description
                  "Opcode-related configuration";

                uses pcp:opcode;
            }

            container option-configuration {
                description
                 "Option-related configuration";

                uses option-server;
            }

           leaf port-randomization-enable {
              type boolean;
              description
                "Enable/disable port randomization
                  feature.";
           }
```

```
            leaf port-preservation-enable {
               type boolean;
               description
                 "Indicates whether the PCP server should
                   preserve the internal port number.";
            }

            leaf port-parity-preservation-enable {
               type boolean;
               description
                 "Indicates whether the PCP server should
                   preserve the port parity of the
                   internal port number.";
            }

            leaf nonce-validation-checks-enable {
               type boolean;
               description
                 "Indicates whether the PCP server has to
                   disable/enable Nonce validation checks.";
            }

            leaf subscriber-mask {
               type uint8 {
                  range "0 .. 128";
               }
            description
             "The subscriber-mask is an integer that indicates
              the length of significant bits to be applied on
              the source IPv6 address (internal side) to
              identify unambiguously a CPE.

              Subscriber-mask is a system-wide configuration
              parameter that is used to enforce generic per-subscriber
              policies (e.g., port-quota).

              Applying these generic policies does not require
              configuring every subscriber's prefix.

              Example: suppose the 2001:db8:100:100::/56 prefix is
              assigned to a DS-Lite enabled CPE. Suppose also that the
              2001:db8:100:100::1 is the IPv6 address used by the
              client that resides in that CPE. When the server
              receives  a packet from this client,
              the server applies the subscriber-mask (e.g., 56) on
              the source IPv6 address to compute the associated prefix
              for this client (that is 2001:db8:100:100::/56).  Then,
              the server enforces policies based on that prefix
```

```
              (2001:db8:100:100::/56), not on the exact
              source IPv6 address.";
            }

            leaf port-quota {
               type uint16;
               description
                 "configure a port quota to be assigned per
                   PCP client/subscriber.";
            }

            list exclude-ports {
                 key "id";
                 description
                    "The set of ports not to be assigned
                      by the server.";

                 leaf id {
                     type uint16;
                     description
                       "An identifier";
                 }

                 uses pcp:port-number;
            }

            list protocol {
                 key "protocol-id";
                 description
                   "set of protocols supported by
                     the PCP-controlled function.";

                 leaf protocol-id {
                 type uint8;
                 description
                  "identifier of the protocol";
                 }
            }

            leaf epoch-set {
               type uint32;
               description
                 "Set the Epoch parameter.";
            }

            container lifetime {
               description
                "Configure values for the lifetime to be
```

```
                assigned to requesting PCP clients.

               The client requests a certain lifetime, and the server
               responds with the assigned lifetime.

               The server may grant a lifetime smaller or larger than
               the requested lifetime.

               The minimum value should be 120 seconds.

               The maximum value should be the remaining
               lifetime of the IP address assigned to
               the PCP client if that information is available,
               or half the lifetime of IP address
               assignments, or 24 hours.

               Excessively long lifetimes can cause consumption
               of ports even if the internal host is no longer
               interested in receiving the traffic or is no
               longer connected to the network.
               (Section 15 [RFC6877].";

                leaf minimum-lifetime {
                    type uint32;
                    default 120;
                    description
                      "Minimum lifetime.";
                }

                leaf maximum-lifetime {
                    type uint32;
                    default 86400;
                    description
                     "Maximum lifetime.";
                }
            }

            container error-lifetime {
               description
                "Configure values for the error lifetime to be
                 returned to requesting PCP clients.";

                leaf minimum-error-lifetime {
                    type uint32;
                    default 30;
                    description
                      "Minimum error lifetime, in seconds.
```

```
                        [RFC6877] recommends that short lifetime
                        errors use a 30-second lifetime.";
                }

                leaf maximum-error-lifetime {
                    type uint32;
                    default 1800;
                    description
                     "Maximum error lifetime, in seconds.

                     [RFC6877] recommends that long lifetime
                     errors use a 30-minute lifetime.";
                }
            }

            container mapping-table {
               description
                 "PCP mapping table as maintained by
                   the PCP server";

                list mapping-entry {
                  key "index";
                  description
                    "PCP mapping entry";
                  uses pcp:mapping-entry;
                }
             }


        container traffic-statistics {

            description
             "traffic statistics";

            uses pcp:traffic-stat;

            container mapping-table {
                description
                    "mapping table statistics";

                uses pcp:mapping-table-stats;
            }

            leaf port-in-use {
                 type percent;
                 description
                  "ratio of the port usage.";
            }
```

```
          }
        }
      }
    }
  }
```

<CODE ENDS>

4. 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 support of SSH is mandatory to implement secure transport [RFC6242]. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and contents.

There is a number of data nodes defined in the YANG module which can, be created, modified and deleted (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) applied to these data nodes without proper protection can negatively affect network operations. In particular, configuring a fake PCP server may ve used to redirect the traffic from a PCP client to an illegitimate server.

5. IANA Considerations

This document requests IANA to register the following URIs in the "IETF XML Registry" [RFC3688]:

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

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

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

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

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

This document requests IANA to register the following YANG modules in the "YANG Module Names" registry [RFC7950].

```
name: ietf-pcp
namespace: urn:ietf:params:xml:ns:yang:ietf-pcp
prefix: pcp
reference: RFC XXXX

name: ietf-pcp-client
namespace: urn:ietf:params:xml:ns:yang:ietf-pcp-client
prefix: pcp-client
reference: RFC XXXX

name: ietf-pcp-iwf
namespace: urn:ietf:params:xml:ns:yang:ietf-pcp-iwf
prefix: pcp-iwf
reference: RFC XXXX

name: ietf-pcp-proxy
namespace: urn:ietf:params:xml:ns:yang:ietf-pcp-proxy
prefix: pcp-proxy
reference: RFC XXXX

name: ietf-pcp-server
namespace: urn:ietf:params:xml:ns:yang:ietf-pcp-server
prefix: pcp-server
reference: RFC XXXX
```

6. References

6.1. Normative references

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

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

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

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

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

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <https://www.rfc-editor.org/info/rfc6887>.

[RFC6970] Boucadair, M., Penno, R., and D. Wing, "Universal Plug and Play (UPnP) Internet Gateway Device - Port Control Protocol Interworking Function (IGD-PCP IWF)", RFC 6970, DOI 10.17487/RFC6970, July 2013, <https://www.rfc-editor.org/info/rfc6970>.

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

[RFC7220] Boucadair, M., Penno, R., and D. Wing, "Description Option for the Port Control Protocol (PCP)", RFC 7220, DOI 10.17487/RFC7220, May 2014, <https://www.rfc-editor.org/info/rfc7220>.

[RFC7225] Boucadair, M., "Discovering NAT64 IPv6 Prefixes Using the Port Control Protocol (PCP)", RFC 7225, DOI 10.17487/RFC7225, May 2014, <https://www.rfc-editor.org/info/rfc7225>.

[RFC7291] Boucadair, M., Penno, R., and D. Wing, "DHCP Options for the Port Control Protocol (PCP)", RFC 7291, DOI 10.17487/RFC7291, July 2014, <https://www.rfc-editor.org/info/rfc7291>.

[RFC7488] Boucadair, M., Penno, R., Wing, D., Patil, P., and T. Reddy, "Port Control Protocol (PCP) Server Selection", RFC 7488, DOI 10.17487/RFC7488, March 2015, <https://www.rfc-editor.org/info/rfc7488>.

[RFC7648] Perreault, S., Boucadair, M., Penno, R., Wing, D., and S. Cheshire, "Port Control Protocol (PCP) Proxy Function", RFC 7648, DOI 10.17487/RFC7648, September 2015, <https://www.rfc-editor.org/info/rfc7648>.

[RFC7652] Cullen, M., Hartman, S., Zhang, D., and T. Reddy, "Port Control Protocol (PCP) Authentication Mechanism", RFC 7652, DOI 10.17487/RFC7652, September 2015, <https://www.rfc-editor.org/info/rfc7652>.

[RFC7753] Sun, Q., Boucadair, M., Sivakumar, S., Zhou, C., Tsou, T., and S. Perreault, "Port Control Protocol (PCP) Extension for Port-Set Allocation", RFC 7753, DOI 10.17487/RFC7753, February 2016, <https://www.rfc-editor.org/info/rfc7753>.

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

6.2. Informative references

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

Authors' Addresses

Mohamed Boucadair
Orange
Rennes 35000
France

EMail: mohamed.boucadair@orange.com

Christian Jacquenet
Orange
Rennes 35000
France

EMail: christian.jacquenet@orange.com

Senthil Sivakumar
Cisco Systems
7100-8 Kit Creek Road
Research Triangle Park, North Carolina 27709
USA

Phone: +1 919 392 5158
EMail: ssenthil@cisco.com

Suresh Vinapamula
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, CA 94089
USA

Phone: +1 408 936 5441
EMail: sureshk@juniper.net