

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 26, 2017

M. Boucadair
C. Jacquenet
Orange
S. Sivakumar
Cisco Systems
S. Vinapamula
Juniper Networks
November 22, 2016

**YANG Data Models for the Port Control Protocol (PCP)
draft-boucadair-pcp-yang-03**

Abstract

This document defines YANG data models 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.

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 May 26, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
1.1.	Requirements Language	3
1.2.	Tree Diagrams	3
1.3.	IP Address Format	4
2.	Overview of the PCP Data Models	4
2.1.	Common PCP	4
2.2.	PCP Client	4
2.3.	UPnP IGD/PCP Interworking Function	8
2.4.	PCP Proxy	11
2.5.	PCP Server	15
3.	YANG Modules	21
3.1.	Common PCP Module	21
3.2.	PCP Client	38
3.3.	UPnP IGD/PCP Interworking Function	44
3.4.	PCP Proxy	50
3.5.	PCP Server	58
4.	Security Considerations	74
5.	IANA Considerations	75
6.	References	76
6.1.	Normative references	76
6.2.	Informative references	78
	Authors' Addresses	78

[1.](#) Introduction

This document defines a data model for the Port Control Protocol (PCP, [[RFC6887](#)]) using the YANG data modeling language [[RFC6020](#)]. The following functional elements are in scope:

- o PCP client [[RFC6887](#)].
- o PCP server [[RFC6887](#)].
- o PCP proxy [[RFC7648](#)].
- o 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:

- o PCP Description option [[RFC7220](#)].
- o PCP Prefix64 discovery option [[RFC7225](#)].
- o 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 data models is defined in [[RFC6020](#)].

1.2. Tree Diagrams

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

- o Brackets "[" and "]" enclose list keys.
- o Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- o Abbreviations before data node names: "rw" means configuration (read-write), "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" a container with presence, and "*" denotes a "list" or "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

- o 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 Data Models

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

2.1. Common PCP

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

2.2. PCP Client

Figure 1 depicts the YANG data model for the PCP client.

```

module: ietf-pcp-client
  +--rw pcp-client-config
  |   +--rw enable?                boolean
  |   +--rw description?          string
  |   +--rw pcp-client-instances
  |   |   +--rw pcp-client-instance* [id]
  |   |   |   +--rw id                uint32
  |   |   |   +--rw name?            string
  |   |   |   +--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-family    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 (port-type)?
|         | | +--:(single-port-number)
|         | | | +--rw single-port-number?  inet:port-number
|         | | +--:(port-range)
|         | | | +--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 (port-type)?
|         | | +--:(single-port-number)
|         | | | +--rw single-port-number?  inet:port-number
|         | | +--:(port-range)
|         | | | +--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
+--ro pcpc-client-state
+--ro pcpc-client-instances
+--ro pcpc-client-instance* [id]
+--ro id                        int32
+--ro name?                    string
+--ro pcpc-client-ip-address* [address-id]

```



```

| +--ro address-id          uint32
| +--ro ip-address?   inet:ipv6-address
+--ro supported-version* [version]
| +--ro version          uint8
+--ro preferred-version?   uint8
+--ro pcp-server-address* [pcp-server-id]
| +--ro pcp-server-id          uint32
| +--ro pcp-server-ip-address* [address-id]
| | +--ro address-id          uint32
| | +--ro ip-address          inet:ipv6-address
| +--ro external-address-family    inet:ip-version
| +--ro stale-external-ip-address?  inet:ipv6-prefix
| +--ro source?                    enumeration
| +--ro in-use?                    boolean
| +--ro server-epoch?              uint32
| +--ro client-epoch?              uint32
| +--ro current-version?           uint8
+--ro authentication-support?  boolean
+--ro opcode-capability
| +--ro map?          boolean
| +--ro peer?         boolean
| +--ro announce?     boolean
+--ro option-capability
| +--ro third-party?   boolean
| +--ro prefer-failure? boolean
| +--ro filter
| | +--ro filter-enabled          boolean
| | +--ro max-filters?           uint32
| +--ro port-set?              boolean
| +--ro description
| | +--ro description-enabled          boolean
| | +--ro max-description?             uint32
| +--ro prefix64?               boolean
+--ro opcode-configuration
| +--ro map?          boolean
| +--ro peer?         boolean
| +--ro announce?     boolean
+--ro option-configuration
| +--ro third-party?   boolean
| +--ro prefer-failure? boolean
| +--ro filter
| | +--ro filter-enabled          boolean
| | +--ro max-filters?           uint32
| +--ro port-set?              boolean
| +--ro description
| | +--ro description-enabled          boolean
| | +--ro max-description?             uint32
| +--ro prefix64?               boolean

```



```

+--ro authentication-enabled?  boolean
+--ro mapping-table
| +--ro mapping-entry* [index]
|   +--ro index                uint32
|   +--ro status?              enumeration
|   +--ro mapping-nonce        string
|   +--ro internal-ip-address   inet:ipv6-prefix
|   +--ro internal-port
|   | +--ro (port-type)?
|   | | +--:(single-port-number)
|   | | | +--ro single-port-number?  inet:port-number
|   | | +--:(port-range)
|   | | | +--ro start-port-number?    inet:port-number
|   | | | +--ro end-port-number?      inet:port-number
|   +--ro external-ip-address     inet:ipv6-prefix
|   +--ro external-port
|   | +--ro (port-type)?
|   | | +--:(single-port-number)
|   | | | +--ro single-port-number?  inet:port-number
|   | | +--:(port-range)
|   | | | +--ro start-port-number?    inet:port-number
|   | | | +--ro end-port-number?      inet:port-number
|   +--ro protocol                uint8
|   +--ro lifetime                uint32
|   +--ro third-party-address?     inet:ipv6-prefix
|   +--ro filter* [filter-id]
|   | +--ro filter-id              uint32
|   | +--ro remote-ip-prefix       inet:ipv6-prefix
|   | +--ro remote-port-number     inet:port-number
|   +--ro description?            string
|   +--ro prefer-failure-tagged?   boolean
|   +--ro status-code?            enumeration
+--ro traffic-statistics
  +--ro traffic-statistics
  | +--ro sent-packet?             yang:zero-based-counter64
  | +--ro sent-byte?              yang:zero-based-counter64
  | +--ro rcvd-packet?            yang:zero-based-counter64
  | +--ro rcvd-byte?              yang:zero-based-counter64
  | +--ro dropped-packet?         yang:zero-based-counter64
  | +--ro dropped-byte?           yang:zero-based-counter64
  +--ro opcode-statistics
  | +--ro sent-map?               yang:zero-based-counter64
  | +--ro rcvd-map?              yang:zero-based-counter64
  | +--ro sent-peer?             yang:zero-based-counter64
  | +--ro rcvd-peer?             yang:zero-based-counter64
  | +--ro sent-announce?         yang:zero-based-counter64
  | +--ro rcvd-announce?         yang:zero-based-counter64
  | +--ro rcvd-unknown?          yang:zero-based-counter64

```



```

|   +--ro rcvd-malformed?   yang:zero-based-counter64
+--ro mapping-table
|   +--ro current-mt-size?   yang:zero-based-counter64
|   +--ro max-mt-size?       uint32

```

Figure 1: PCP Client YANG Data Model

2.3. UPnP IGD/PCP Interworking Function

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

```

module: ietf-pcp-iwf
+--rw pcp-iwf-config
|   +--rw enable?           boolean
|   +--rw pcp-igd-iwf-instances
|       +--rw pcp-igd-iwf-instance* [id]
|           +--rw id                uint32
|           +--rw name?             string
|           +--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-family inet:ip-version
|           |   +--rw stale-external-ip-address? inet:ipv6-prefix
|       +--rw authentication-enable? boolean
|       +--rw igd-version
|           | +--rw igd-version? enumeration
|       +--rw mapping-table
|           +--rw mapping-entry* [index]
|               +--rw igd-control-point-address? inet:ip-address
|               +--rw igd-control-point-port?   inet:port-number
|               +--rw index                     uint32
|               +--rw status?                   enumeration
|               +--rw mapping-nonce             string
|               +--rw internal-ip-address        inet:ipv6-prefix
|               +--rw internal-port
|                   | +--rw (port-type)?
|                   |   +--:(single-port-number)
|                   |   | +--rw single-port-number? inet:port-number
|                   |   +--:(port-range)
|                   |       +--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 (port-type)?
|         |   +--:(single-port-number)
|         |     | +--rw single-port-number?   inet:port-number
|         |     +--:(port-range)
|         |       +--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
+--ro pcpiwf-state
  +--ro pcpiigd-iwf-instances
    +--ro pcpiigd-iwf-instance* [id]
      +--ro id                             int32
      +--ro name?                          string
      +--ro supported-version* [version]
        | +--ro version                     uint8
      +--ro preferred-version?              uint8
      +--ro pcpiigd-iwf-ip-address* [address-id]
        | +--ro address-id                  uint32
        | +--ro ip-address?                 inet:ipv6-address
      +--ro authentication-support?         boolean
      +--ro authentication-enabled?         boolean
      +--ro igd-version-capability
        | +--ro igd-version?                enumeration
      +--ro enabled-igd-version
        | +--ro igd-version?                enumeration
      +--ro pcpi-server-address* [pcpi-server-id]
        | +--ro pcpi-server-id              uint32
        | +--ro pcpi-server-ip-address* [address-id]
        |   | +--ro address-id              uint32
        |   | +--ro ip-address              inet:ipv6-address
        | +--ro external-address-family    inet:ip-version
        | +--ro stale-external-ip-address? inet:ipv6-prefix
        | +--ro source?                    enumeration
        | +--ro in-use?                     boolean
        | +--ro server-epoch?               uint32
        | +--ro client-epoch?               uint32
        | +--ro current-version?            uint8
      +--ro mapping-table
        | +--ro mapping-entry* [index]
        |   +--ro index                     uint32
        |   +--ro status?                   enumeration

```



```

|   +--ro mapping-nonce          string
|   +--ro internal-ip-address    inet:ipv6-prefix
|   +--ro internal-port
|   |   +--ro (port-type)?
|   |   |   +--:(single-port-number)
|   |   |   |   +--ro single-port-number?    inet:port-number
|   |   |   +--:(port-range)
|   |   |   |   +--ro start-port-number?    inet:port-number
|   |   |   |   +--ro end-port-number?      inet:port-number
|   +--ro external-ip-address    inet:ipv6-prefix
|   +--ro external-port
|   |   +--ro (port-type)?
|   |   |   +--:(single-port-number)
|   |   |   |   +--ro single-port-number?    inet:port-number
|   |   |   +--:(port-range)
|   |   |   |   +--ro start-port-number?    inet:port-number
|   |   |   |   +--ro end-port-number?      inet:port-number
|   +--ro protocol                uint8
|   +--ro lifetime                uint32
|   +--ro third-party-address?    inet:ipv6-prefix
|   +--ro filter* [filter-id]
|   |   +--ro filter-id            uint32
|   |   +--ro remote-ip-prefix    inet:ipv6-prefix
|   |   +--ro remote-port-number  inet:port-number
|   +--ro description?           string
|   +--ro prefer-failure-tagged?  boolean
|   +--ro status-code?           enumeration
|   +--ro igd-control-point-address?  inet:ip-address
|   +--ro igd-control-point-port?  inet:port-number
+--ro traffic-statistics
  +--ro traffic-statistics
    |   +--ro sent-packet?        yang:zero-based-counter64
    |   +--ro sent-byte?         yang:zero-based-counter64
    |   +--ro rcvd-packet?       yang:zero-based-counter64
    |   +--ro rcvd-byte?        yang:zero-based-counter64
    |   +--ro dropped-packet?    yang:zero-based-counter64
    |   +--ro dropped-byte?     yang:zero-based-counter64
  +--ro opcode-statistics
    |   +--ro sent-map?          yang:zero-based-counter64
    |   +--ro rcvd-map?          yang:zero-based-counter64
    |   +--ro sent-peer?         yang:zero-based-counter64
    |   +--ro rcvd-peer?         yang:zero-based-counter64
    |   +--ro sent-announce?     yang:zero-based-counter64
    |   +--ro rcvd-announce?     yang:zero-based-counter64
    |   +--ro rcvd-unknown?      yang:zero-based-counter64
    |   +--ro rcvd-malformed?    yang:zero-based-counter64
  +--ro mapping-table
    +--ro current-mt-size?      yang:zero-based-counter64

```



```

+--ro max-mt-size?      uint32

```

Figure 2: IWF YANG Data Model

2.4. PCP Proxy

Figure 3 depicts the YANG data model for the PCP proxy.

```

module: ietf-pcp-proxy
+--rw pcproxy-config
| +--rw enable?          boolean
| +--rw description?     string
| +--rw pcproxy-instances
| | +--rw pcproxy-instance* [id]
| | | +--rw id          uint32
| | | +--rw name?       string
| | | +--rw version* [version]
| | | | +--rw version    uint8
| | | +--rw pc-servers* [pc-server-id]
| | | | +--rw pc-server-id    uint32
| | | | +--rw pc-server-ip-address* [address-id]
| | | | | +--rw address-id    uint32
| | | | | +--rw ip-address    inet:ip6-address
| | | | +--rw external-address-family    inet:ip-version
| | | | +--rw stale-external-ip-address? inet:ip6-prefix
| | | +--rw authentication-enable? boolean
| | | +--rw opcode-configuration
| | | | +--rw map?          boolean
| | | | +--rw peer?         boolean
| | | | +--rw announce?     boolean
| | | | +--rw relay-unknown? 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 relay-mandatory-unknown-option? boolean
| | | +--rw relay-optionnal-unknown-option? boolean
+--rw terminate-proxy-recursion? 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 (port-type)?
|         |   |   +--:(single-port-number)
|         |   |   |   +--rw single-port-number?    inet:port-number
|         |   |   +--:(port-range)
|         |   |   |   +--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 (port-type)?
|         |   |   +--:(single-port-number)
|         |   |   |   +--rw single-port-number?    inet:port-number
|         |   |   +--:(port-range)
|         |   |   |   +--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 local-assigned-ip-address?           inet:ipv6-prefix
|         +--rw local-assigned-port
|         |   +--rw (port-type)?
|         |   |   +--:(single-port-number)
|         |   |   |   +--rw single-port-number?    inet:port-number
|         |   |   +--:(port-range)
|         |   |   |   +--rw start-port-number?     inet:port-number
|         |   |   |   +--rw end-port-number?       inet:port-number
+--ro pcproxy-state
  +--ro pcproxy-instances
    +--ro pcproxy-instance* [id]
      +--ro id                                     int32
      +--ro name?                                 string
      +--ro supported-version* [version]
      |   +--ro version                           uint8
      +--ro preferred-version?                     uint8
      +--ro pcproxy-ip-address* [address-id]
      |   +--ro address-id                         uint32
      |   +--ro pcproxy-ip-address?                inet:ipv6-address
      +--ro pcproxy-server-address* [pcproxy-server-id]

```



```

| +--ro pcp-server-id                uint32
| +--ro pcp-server-ip-address* [address-id]
| | +--ro address-id                uint32
| | +--ro ip-address                inet:ipv6-address
| +--ro external-address-family      inet:ip-version
| +--ro stale-external-ip-address?   inet:ipv6-prefix
| +--ro source?                      enumeration
| +--ro in-use?                      boolean
| +--ro server-epoch?               uint32
| +--ro client-epoch?               uint32
| +--ro current-version?             uint8
+--ro authentication-support?         boolean
+--ro pcp-controlled-function-capability
| +--ro nat44?                      boolean
| +--ro nat64?                      boolean
| +--ro ds-lite?                    boolean
| +--ro nptv6?                      boolean
| +--ro ipv4-firewall?              boolean
| +--ro ipv6-firewall?              boolean
| +--ro port-range-router?          boolean
+--ro opcode-capability
| +--ro map?                        boolean
| +--ro peer?                      boolean
| +--ro announce?                  boolean
| +--ro relay-unknown?              boolean
+--ro option-capability
| +--ro third-party?                boolean
| +--ro prefer-failure?              boolean
| +--ro filter
| | +--ro filter-enabled              boolean
| | +--ro max-filters?               uint32
| +--ro port-set?                  boolean
| +--ro description
| | +--ro description-enabled          boolean
| | +--ro max-description?            uint32
| +--ro prefix64?                  boolean
| +--ro relay-mandatory-unknown-option? boolean
| +--ro relay-optionnal-unknown-option? boolean
+--ro opcode-configuration
| +--ro map?                        boolean
| +--ro peer?                      boolean
| +--ro announce?                  boolean
+--ro option-configuration
| +--ro third-party?                boolean
| +--ro prefer-failure?              boolean
| +--ro filter
| | +--ro filter-enabled              boolean
| | +--ro max-filters?               uint32

```



```

| +--ro port-set?                               boolean
| +--ro description
| | +--ro description-enabled                   boolean
| | +--ro max-description?                     uint32
| +--ro prefix64?                             boolean
| +--ro relay-mandatory-unknown-option?       boolean
| +--ro relay-optionnal-unknown-option?       boolean
+--ro authentication-enabled?                 boolean
+--ro terminate-proxy-recursion-status?       boolean
+--ro mapping-table
| +--ro mapping-entry* [index]
| | +--ro index                               uint32
| | +--ro status?                           enumeration
| | +--ro mapping-nonce                     string
| | +--ro internal-ip-address               inet:ipv6-prefix
| | +--ro internal-port
| | | +--ro (port-type)?
| | | | +--:(single-port-number)
| | | | | +--ro single-port-number?         inet:port-number
| | | | +--:(port-range)
| | | | | +--ro start-port-number?          inet:port-number
| | | | | +--ro end-port-number?            inet:port-number
| | +--ro external-ip-address               inet:ipv6-prefix
| | +--ro external-port
| | | +--ro (port-type)?
| | | | +--:(single-port-number)
| | | | | +--ro single-port-number?         inet:port-number
| | | | +--:(port-range)
| | | | | +--ro start-port-number?          inet:port-number
| | | | | +--ro end-port-number?            inet:port-number
| | +--ro protocol                         uint8
| | +--ro lifetime                         uint32
| | +--ro third-party-address?              inet:ipv6-prefix
| | +--ro filter* [filter-id]
| | | +--ro filter-id                       uint32
| | | +--ro remote-ip-prefix                inet:ipv6-prefix
| | | +--ro remote-port-number              inet:port-number
| | +--ro description?                     string
| | +--ro prefer-failure-tagged?            boolean
| | +--ro local-assigned-ip-address?        inet:ipv6-prefix
| | +--ro local-assigned-port
| | | +--ro (port-type)?
| | | | +--:(single-port-number)
| | | | | +--ro single-port-number?         inet:port-number
| | | | +--:(port-range)
| | | | | +--ro start-port-number?          inet:port-number
| | | | | +--ro end-port-number?            inet:port-number
| | +--ro status-code?                     enumeration

```



```

+--ro traffic-statistics
  +--ro client-facing-interface
    | +--ro traffic-statistics
    | | +--ro sent-packet?      yang:zero-based-counter64
    | | +--ro sent-byte?       yang:zero-based-counter64
    | | +--ro rcvd-packet?     yang:zero-based-counter64
    | | +--ro rcvd-byte?       yang:zero-based-counter64
    | | +--ro dropped-packet?   yang:zero-based-counter64
    | | +--ro dropped-byte?     yang:zero-based-counter64
    | +--ro opcode-statistics
    |   +--ro sent-map?        yang:zero-based-counter64
    |   +--ro rcvd-map?        yang:zero-based-counter64
    |   +--ro sent-peer?       yang:zero-based-counter64
    |   +--ro rcvd-peer?       yang:zero-based-counter64
    |   +--ro sent-announce?   yang:zero-based-counter64
    |   +--ro rcvd-announce?   yang:zero-based-counter64
    |   +--ro rcvd-unknown?     yang:zero-based-counter64
    |   +--ro rcvd-malformed?   yang:zero-based-counter64
  +--ro server-facing-interface
    | +--ro traffic-statistics
    | | +--ro sent-packet?      yang:zero-based-counter64
    | | +--ro sent-byte?       yang:zero-based-counter64
    | | +--ro rcvd-packet?     yang:zero-based-counter64
    | | +--ro rcvd-byte?       yang:zero-based-counter64
    | | +--ro dropped-packet?   yang:zero-based-counter64
    | | +--ro dropped-byte?     yang:zero-based-counter64
    | +--ro opcode-statistics
    |   +--ro sent-map?        yang:zero-based-counter64
    |   +--ro rcvd-map?        yang:zero-based-counter64
    |   +--ro sent-peer?       yang:zero-based-counter64
    |   +--ro rcvd-peer?       yang:zero-based-counter64
    |   +--ro sent-announce?   yang:zero-based-counter64
    |   +--ro rcvd-announce?   yang:zero-based-counter64
    |   +--ro rcvd-unknown?     yang:zero-based-counter64
    |   +--ro rcvd-malformed?   yang:zero-based-counter64
  +--ro mapping-table
    +--ro current-mt-size?     yang:zero-based-counter64
    +--ro max-mt-size?         uint32

```

Figure 3: PCP Proxy YANG Data Model

2.5. PCP Server

Figure 4 depicts the YANG data model for the PCP server.

```

module: ietf-pcp-server
  +--rw pcp-server-config

```



```

|  +--rw enable?                boolean
|  +--rw pcp-server-instances
|    +--rw pcp-server-instance* [id]
|      +--rw id                  uint32
|      +--rw name?              string
|      +--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 (port-type)?
|          +--:(single-port-number)
|            +--rw single-port-number?  inet:port-number

```



```

|         |         +--:(port-range)
|         |         +--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 (port-type)?
|         |   |   |   |   +--:(single-port-number)
|         |   |   |   |   |   +--rw single-port-number?    inet:port-number
|         |   |   |   |   +--:(port-range)
|         |   |   |   |   |   +--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 (port-type)?
|         |   |   |   +--:(single-port-number)
|         |   |   |   |   +--rw single-port-number?    inet:port-number
|         |   |   |   +--:(port-range)
|         |   |   |   |   +--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
|         +--ro pcg-server-state
|         |   +--ro pcg-server-instances
|         |   |   +--ro pcg-server-instance* [id]
|         |   |   |   +--ro id                  int32
|         |   |   |   +--ro name?                string
|         |   |   +--ro supported-version* [version]

```



```
| +--ro version                uint8
+--ro preferred-version?       uint8
+--ro configured-pcp-server-ip-address* [address-id]
| +--ro address-id             uint32
| +--ro ip-address?    inet:ipv6-address
+--ro external-ip-address-pool* [address-id]
| +--ro address-id             uint32
| +--ro external-ip-pool?    inet:ipv6-prefix
+--ro authentication-support?   boolean
+--ro opcode-capability
| +--ro map?                   boolean
| +--ro peer?                  boolean
| +--ro announce?             boolean
+--ro option-capability
| +--ro third-party?           boolean
| +--ro prefer-failure?        boolean
| +--ro filter
| | +--ro filter-enabled           boolean
| | +--ro max-filters?            uint32
| +--ro port-set?              boolean
| +--ro description
| | +--ro description-enabled      boolean
| | +--ro max-description?        uint32
| +--ro prefix64?              boolean
+--ro port-randomization-support? boolean
+--ro port-preservation-suport?  boolean
+--ro port-parity-preservation-support? boolean
+--ro protocol-capabilities* [protocol-id]
| +--ro protocol-id            uint8
+--ro pcp-controlled-function-capability
| +--ro nat44?                  boolean
| +--ro nat64?                  boolean
| +--ro ds-lite?                boolean
| +--ro nptv6?                  boolean
| +--ro ipv4-firewall?          boolean
| +--ro ipv6-firewall?          boolean
| +--ro port-range-router?      boolean
+--ro opcode-configuration
| +--ro map?                    boolean
| +--ro peer?                  boolean
| +--ro announce?              boolean
+--ro option-configuration
| +--ro third-party?           boolean
| +--ro prefer-failure?        boolean
| +--ro filter
| | +--ro filter-enabled           boolean
| | +--ro max-filters?            uint32
| +--ro port-set-option
```



```

| | +--ro port-set-enable          boolean
| | +--ro default-port-set-size?   uint16
| | +--ro maximum-port-set-size?   uint16
| +--ro description
| | +--ro description-enabled       boolean
| | +--ro max-description?          uint32
| +--ro prefix64-option
| | +--ro prefix64-option-enable?   boolean
| | +--ro prefix64* [prefix64-id]
| | | +--ro prefix64-id             uint32
| | | +--ro prefix64?               inet:ipv6-prefix
| | | +--ro suffix?                 yang:hex-string
| | | +--ro dest-ipv4-prefix* [ipv4-prefix-id]
| | | | +--ro ipv4-prefix-id        uint32
| | | | +--ro ipv4-prefix?          inet:ipv4-prefix
+--ro authentication-enabled?        boolean
+--ro port-randomization-enable?     boolean
+--ro port-preservation-enable?      boolean
+--ro port-parity-preservation-enable? boolean
+--ro enabled-protocol* [protocol-id]
| +--ro protocol-id                  uint8
+--ro subscriber-mask-support?       boolean
+--ro subscriber-mask?               uint8
+--ro port-quota?                    uint16
+--ro exclude-ports* [id]
| +--ro id                           uint16
| +--ro (port-type)?
| | +--:(single-port-number)
| | | +--ro single-port-number?      inet:port-number
| | +--:(port-range)
| | | +--ro start-port-number?       inet:port-number
| | | +--ro end-port-number?         inet:port-number
+--ro nonce-validation-checks-enable? boolean
+--ro epoch?                         uint32
+--ro lifetime
| +--ro minimum-lifetime?            uint32
| +--ro maximum-lifetime?           uint32
+--ro error-lifetime
| +--ro minimum-error-lifetime?      uint32
| +--ro maximum-error-lifetime?      uint32
+--ro mapping-table
| +--ro mapping-entry* [index]
| | +--ro index                     uint32
| | +--ro status?                   enumeration
| | +--ro mapping-nonce              string
| | +--ro internal-ip-address         inet:ipv6-prefix
| | +--ro internal-port
| | | +--ro (port-type)?

```



```

|   |   +--:(single-port-number)
|   |   |   +--ro single-port-number?   inet:port-number
|   |   +--:(port-range)
|   |       +--ro start-port-number?   inet:port-number
|   |       +--ro end-port-number?     inet:port-number
|   +--ro external-ip-address          inet:ipv6-prefix
|   +--ro external-port
|   |   +--ro (port-type)?
|   |   +--:(single-port-number)
|   |   |   +--ro single-port-number?   inet:port-number
|   |   +--:(port-range)
|   |       +--ro start-port-number?   inet:port-number
|   |       +--ro end-port-number?     inet:port-number
|   +--ro protocol                     uint8
|   +--ro lifetime                     uint32
|   +--ro third-party-address?         inet:ipv6-prefix
|   +--ro filter* [filter-id]
|   |   +--ro filter-id                uint32
|   |   +--ro remote-ip-prefix         inet:ipv6-prefix
|   |   +--ro remote-port-number       inet:port-number
|   +--ro description?                 string
|   +--ro prefer-failure-tagged?       boolean
|   +--ro status-code?                 enumeration
+--ro traffic-statistics
  +--ro traffic-statistics
    |   +--ro sent-packet?              yang:zero-based-counter64
    |   +--ro sent-byte?                yang:zero-based-counter64
    |   +--ro rcvd-packet?              yang:zero-based-counter64
    |   +--ro rcvd-byte?                yang:zero-based-counter64
    |   +--ro dropped-packet?           yang:zero-based-counter64
    |   +--ro dropped-byte?             yang:zero-based-counter64
  +--ro opcode-statistics
    |   +--ro sent-map?                 yang:zero-based-counter64
    |   +--ro rcvd-map?                 yang:zero-based-counter64
    |   +--ro sent-peer?                yang:zero-based-counter64
    |   +--ro rcvd-peer?                yang:zero-based-counter64
    |   +--ro sent-announce?            yang:zero-based-counter64
    |   +--ro rcvd-announce?            yang:zero-based-counter64
    |   +--ro rcvd-unknown?             yang:zero-based-counter64
    |   +--ro rcvd-malformed?           yang:zero-based-counter64
  +--ro mapping-table
    |   +--ro current-mt-size?          yang:zero-based-counter64
    |   +--ro max-mt-size?              uint32
  +--ro port-in-use?                   percent

```

Figure 4: PCP Server YANG Data Model

3. YANG Modules

3.1. Common PCP Module

```
<CODE BEGINS> file "ietf-pcp@2015-08-05.yang"
module ietf-pcp {
  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) 2016 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 2015-08-05 {
    description "Changes tbc.";
    reference "tbc";
  }

  /*
   * Grouping
   */

  //Description option

  grouping description-option {
    description
```



```
        "used to configure DESCRIPTION option [RFC7220].";

    leaf description-enabled {
        type boolean;
        description
            "Enable/disable DESCRIPTION option.";
    }

    leaf max-description {
        type uint32;
        description
            "Indicates the maximum length of the description
             associated with a mapping.";
    }
}

//Filter option

grouping filter-option {
    description
        "FILTER option as defined in [RFC6887].";

    leaf filter-enabled {
        type boolean;
        description
            "Enable/disable FILTER option.";
    }

    leaf max-filters {
        type uint32;
        description
            "Indicates the maximum number of filters
             associated with a mapping.";
    }
}

// Port set option

grouping port-set-option {
    description
        "PORT_SET option [RFC7753].";

    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.";
    }
}

//Opcodes

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

    leaf map {
        type boolean;
        description
            "MAP opcode";
    }

    leaf peer {
        type boolean;
        description
            "PEER opcode";
    }

    leaf announce {
        type boolean;
        description
            "ANNOUNCE opcode.";
    }
}

//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 [RFC6887].";
    }

    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 \[RFC6887\].";
    }

    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 [RFC7220].";
        uses description-option;
    }

    leaf prefix64 {
        type boolean;
        description
            "PREFIX64 PCP option [RFC7225].";
    }
}

// port numbers: single or port range
```



```
grouping port-number {
  description
    "individual port or a range of ports.";

  choice port-type {
    default single-port-number;
    description
      "port type: single or port-range.";

    case single-port-number {
      leaf single-port-number {
        type inet:port-number;
        description
          "used for single port numbers.";
      }
    }

    case port-range {
      leaf start-port-number {
        type inet:port-number;
        description
          "Begining of the port range.";
      }

      leaf end-port-number {
        type inet:port-number;
        description
          "End of the port range.";
      }
    }
  }
}

// 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-family {
    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 pcg-server-address-status {

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

  uses pcg-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 pcpc-controlled-function {
    description
        "A set of PCP-controlled functions.
        One or multiple functions can be controlled
        by the same PCP server. ";

    leaf nat44 {
        type boolean;
        description
            "NAT44";
    }

    leaf nat64 {
        type boolean;
        description
            "NAT64";
    }

    leaf ds-lite {
        type boolean;
        description
            "DS-Lite";
    }

    leaf nptv6 {
        type boolean;
        description
            "NPTv6";
    }
}
```



```
    leaf ipv4-firewall {
        type boolean;
        description
            "IPv4 firewall";
    }

    leaf ipv6-firewall {
        type boolean;
        description
            "IPv6 firewall";
    }

    leaf port-range-router {
        type boolean;
        description
            "Port Range Router";
    }
}

// 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-announce {
```



```
        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@2015-08-05.yang"
module ietf-pcp-client {
  namespace "urn:ietf:params:xml:ns:yang:ietf-pcp-client";
  prefix pcp-client;

  import ietf-inet-types { prefix inet; }
  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) 2016 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 2015-08-05 {
    description "Changes tbc.";
    reference "tbc";
```



```
    }

/*
 *PCP Configuration
 */

    container pcp-client-config {
        description
            "PCP client configuration";

        leaf enable {
            type boolean;
            description
                "Enable/disable the PCP client.";
        }

        leaf description {
            type string;
            description
                "Associated a description with the module.";
        }

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

            list pcp-client-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.";
                }

                list version {
                    key version;
                    description
```



```
        "Indicates the set of supported PCP versions
        (0, 1, 2)";

    uses pcplib:pcplib-version;
}

list pcplib-servers {
    key "pcplib-server-id";
    description
        "List of provisioned PCP servers.";
    uses pcplib:pcplib-server-address;
}

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

container opcode-configuration {
    description
        "Opcode-related configuration";
    uses pcplib:opcode;
}

container option-configuration {
    description
        "Options-related configuration.";
    uses pcplib:option;
}

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

    list mapping-entry {
        key "index";
        description
            "PCP Mapping entry.";
        uses pcplib:mapping-entry;
    }
}

}

}

}

/*
```



```
* PCP state
*/
```

```
container pcp-client-state {

    config false;

    description
        "PCP client state";

    container pcp-client-instances {
        description
            "PCP client instances";

        list pcp-client-instance {

            key "id";

            description
                "PCP client instance";

            leaf id {
                type int32;
                description
                    "PCP client instance identifier.";
            }

            leaf name {
                type string;
                description
                    "A name associated with the PCP client instance.";
            }
        }

        list pcp-client-ip-address {

            key address-id;

            description
                "list of configured PCP client addresses.";

            leaf address-id {
                type uint32;
                description
                    "Address identifier";
            }

            leaf ip-address {
                type inet:ipv6-address;
            }
        }
    }
}
```



```
    description
      "IP address";
  }
}

  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.";
  }

  list pcp-server-address {
    key "pcp-server-id";
    description
      "list of provisioned PCP server.";

    uses pcp:pcp-server-address-status;
  }

  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;
  }

  container opcode-configuration {
    description
```



```
        "Opcode-related configuration.";
        uses pcp:opcode;
    }

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

        uses pcp:option;
    }

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

    container mapping-table {
        description
            "Mapping table";

        list mapping-entry {
            key "index";
            description
                "Mapping entry";

            uses pcp:mapping-entry;
            uses pcp:status-code;
        }
    }

    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@2015-08-05.yang"
module ietf-pcp-iwf {
  namespace "urn:ietf:params:xml:ns:yang:ietf-pcp-iwf";
  prefix pcp-iwf;

  import ietf-inet-types { prefix inet; }
  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
    UPnP IGD/PCP Interworking implementations.

    Copyright (c) 2016 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 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";
}
}

/*
 *PCP Configuration
 */

container pcg-iwf-config {
    description
        "UPnP IGD/PCP Interworking Function";

    leaf enable {
        type boolean;
        description
            "Enable/Disable the UPnP IGD-PCP IWF";
    }

    container pcg-igd-iwf-instances {
        description
            "UPnP IGD/PCP Interworking Function instances";

        list pcg-igd-iwf-instance {
            key "id";

            description
                "UPnP IGD/PCP Interworking Function instance";

            leaf id {
                type uint32;
```



```
        description
            "An identifier of the IWF instance.";
    }

    leaf name {
        type string;
        description
            "A name of the UPnP IGD-PCP IWF instance";
    }

    list version {
        key version;
        description
            "configures one or several PCP versions.";

        uses pcpc:pcp-version;
    }

    list pcpc-servers {
        key "pcpc-server-id";
        description
            "List of configured PCP servers.";
        uses pcpc:pcpc-server-address;
    }

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

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

        uses igd-version;
    }

    container mapping-table {
        description
            "Mapping table as maintained by a
            UPnP IGD-PCP IWF instance";

        list mapping-entry {
            key "index";
            description
                "PCP Mapping Entry.";
        }
    }
}
```



```
        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";
        }
        uses pcplib:mapping-entry;
    }
}

}

}

/*
 * PCP state
 */

container pcplib-iwf-state {

    config false;

    description
        "UPnP IGD/PCP Interworking Function";

    container pcplib-igd-iwf-instances {
        description
            "UPnP IGD/PCP Interworking Function instances";

        list pcplib-igd-iwf-instance {

            key "id";
            description
                "UPnP IGD/PCP Interworking Function instance";

            leaf id {
                type int32;
                description
                    "the identifier of the instance";
            }

            leaf name {
                type string;
                description
```



```
        "the name of the instance";
    }

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

        uses pcpc:pcp-version;
    }

    leaf preferred-version {
        type uint8;
        description
            "Preferred version";
    }

    list pcpc-igd-iwf-ip-address {

        key address-id;

        description
            "local IP addresses of the UPnP IGD-PCP IWF";

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

        leaf ip-address {
            type inet:ipv6-address;
            description
                "An address of the UPnP IGD-PCP IWF";
        }
    }

    leaf authentication-support {
        type boolean;
        description
            "Indicates whether PCP authentication is
            supported.";
    }

    leaf authentication-enabled{
        type boolean;
        description
            "Indicates whether PCP authentication
```



```
        is enabled.";
    }

    container igd-version-capability {
        description
            "List of supported UPnP IGD versions.";

        uses igd-version;
    }

    container enabled-igd-version {
        description
            "Configured UPnP IGD versions";

        uses igd-version;
    }

    list pcg-server-address {
        key "pcg-server-id";
        description
            "List of provisioned PCP servers";

        uses pcg:pcg-server-address-status;
    }

    container mapping-table {
        description
            "PCP Mapping table";

        list mapping-entry {
            key "index";
            description
                "PCP mapping entry.";

            uses pcg:mapping-entry;
            uses pcg:status-code;

            leaf igd-control-point-address {
                type inet:ip-address;
                description
                    "The IP address of a UPnP Control Point";
            }

            leaf igd-control-point-port {
                type inet:port-number;
                description
                    "The port number of a UPnP Control Point";
            }
        }
    }
}
```



```

    }
  }

  container traffic-statistics {

    description
      "traffic statistics";

    uses pcplib:traffic-stat;

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

      uses pcplib:mapping-table-stats;
    }
  }
}
}
}
}
}
<CODE ENDS>

```

3.4. PCP Proxy

```

<CODE BEGINS> file "ietf-pcp-proxy@2015-08-05.yang"
module ietf-pcp-proxy {
  namespace "urn:ietf:params:xml:ns:yang:ietf-pcp-proxy";
  prefix pcplib;

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

  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) 2016 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 2015-08-05 {  
  description "Changes xxxx.";  
  reference "xxxx";  
}
```

```
/*
```

```
*PCP Configuration
```

```
*/
```

```
container pcp-proxy-config {  
  description  
    "PCP proxy";  
  
  leaf enable {  
    type boolean;  
    description  
      "Enable/Disable PCP proxy";  
  }  
}
```

```
leaf description {  
  type string;  
  description  
    "Associated a description with the module.";  
}
```

```
container pcp-proxy-instances {  
  description  
    "PCP proxy instances";  
  
  list pcp-proxy-instance {  
    key "id";  
    description  
      "PCP proxy instance";  
  
    leaf id {  
      type uint32;  
      description  
        "An identifier of the PCP proxy instance";  
    }  
  
    leaf name {
```



```
    type string;
    description
      "A name of the PCP proxy instance";
  }

  list version {
    key version;
    description
      "Supported PCP versions.";
    uses pcplib:pcplib-version;
  }

  list pcplib-servers {
    key "pcplib-server-id";
    description
      "List of provisioned PCP servers.";
    uses pcplib:pcplib-server-address;
  }

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

  container opcode-configuration {
    description
      "Opcode-related configuration";
    uses pcplib:opcode;

    leaf relay-unknown-opcode {
      type boolean;
      description
        "The proxy can be instructed to relay
         or to reject unknown opcodes.";
    }
  }

  container option-configuration {
    description
      "Option-related configuration";
    uses pcplib:option;

    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.";
    }
}

leaf terminate-proxy-recursion {
    type boolean;
    description
        "The proxy can be instructed to terminate
         proxy recursion.";
}

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

    list mapping-entry {
        key "index";
        description
            "PCP mapping entry";

        uses pcp:mapping-entry;

        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;
        }
    }
}
```



```
    }
  }
}

/*
 * PCP state
 */

container pcp-proxy-state {

  config false;

  description
    "PCP proxy";

  container pcp-proxy-instances {
    description
      "PCP proxy Instances";

    list pcp-proxy-instance {

      key "id";
      description
        "PCP proxy Instance";

      leaf id {
        type int32;
        description
          "Identifier";
      }

      leaf name {
        type string;
        description
          "Name of the PCP proxy Instance";
      }

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

        uses pcp:pcp-version;
      }

      leaf preferred-version {
        type uint8;
        description
```



```
        "Configured preferred version";
    }

    list pcp-proxy-ip-address {

        key address-id;

        description
            "List of configured addresses to the
             PCP proxy instance.";

        leaf address-id {
            type uint32;
            description
                "An identifier";
        }

        leaf pcp-proxy-ip-address {
            type inet:ipv6-address;
            description
                "An address";
        }
    }

    list pcp-server-address {
        key "pcp-server-id";
        description
            "list of provisioned PCP servers.";

        uses pcp:pcp-server-address-status;
    }

    leaf authentication-support {
        type boolean;
        description
            "Indicates whether PCP authentication is
             enabled/disabled.";
    }

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

        uses pcp:pcp-controlled-function;
    }

    container opcode-capability {
```



```
    description
      "Opcode-related capabilities.";

    uses pcop:opcode;

    leaf relay-unknown-opcode {
      type boolean;
      description
        "instruction related to the processing of unknown
        opcodes.";
    }
  }

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

    uses pcop:option;

    leaf relay-mandatory-unknown-option {
      type boolean;
      description
        "instruction related to the processing
        of mandatory unknown options.";
    }

    leaf relay-optionnal-unknown-option {
      type boolean;
      description
        "instruction related to the processing
        of optional unknown options.";
    }
  }

  container opcode-configuration {
    description
      "opcode-related configurations.";
    uses pcop:opcode;
  }

  container option-configuration {
    description
      "opcode-related configurations.";

    uses pcop:option;

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



```
        description
            "instruction related to the processing
             of mandatory unknown options.";
    }

    leaf relay-optionnal-unknown-option {
        type boolean;
        description
            "instruction related to the processing
             of optional unknown options.";
    }
}

leaf authentication-enabled {
    type boolean;
    description
        "status of the PCP authentication activation";
}

leaf terminate-proxy-recursion-status {
    type boolean;
    description
        "Indicates whether recursion is
         terminated or not";
}

container mapping-table {
    description
        "mapping table";
    list mapping-entry {
        key "index";
        description
            "mapping entry";
        uses pcpc:mapping-entry;

        leaf local-assigned-ip-address {
            type inet:ipv6-prefix;
            description
                "An address assigned locally by
                 the proxy";
        }

        container local-assigned-port {
            description
                "a port assigned locally by the proxy";

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



```
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) 2016 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 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 pcpc:filter-option;
    }

    container port-set-option {
        description
            "enable/disable PORT_SET option.";

        uses pcpc:port-set-option;
    }

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

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

/*
 * PCP server Configuration
 */

container pcpc-server-config {
    description
        "PCP server";

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

    container pcpc-server-instances {
        description
            "PCP server instances";

        list pcpc-server-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";
    }

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

        uses pcpc-version;
    }

    list pcpc-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 pcop: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 pcplib:pcplib-mapping-entry;
    }
}

}

}

}

/*
 * PCP server State
 */

container pcplib-server-state {

    config false;

    description
        "PCP server";

    container pcplib-server-instances {
        description
            "PCP server instances";

        list pcplib-server-instance {
            key "id";

            description
                "PCP server instance";

            leaf id {
                type int32;
                description
                    "The identifier of the PCP server instance.";
            }

            leaf name {
                type string;
                description
                    "The name of the PCP server instance";
            }

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

                uses pcplib:pcplib-version;
            }
        }
    }
}
```



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

list configured-pcp-server-ip-address {

    key address-id;

    description
        "List of PCP server IP addresses";

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

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

list external-ip-address-pool {

    key address-id;

    description
        "Pool of external IP addresses used to service
         requesting clients.";

    leaf address-id {
        type uint32;
        description
            "An identifier";
    }

    leaf external-ip-pool {
        type inet:ipv6-prefix;
        description
            "An address or prefix";
    }
}
```



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

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

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

  uses pcpc:option;
}

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

leaf port-preservation-support {
  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 pcpc-controlled-function-capability {
    description
        "list of controlled functions.";

    uses pcpc:pcpc-controlled-function;
}

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

    uses pcpc:opcode;
}

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

    uses option-server;
}

leaf authentication-enabled{
    type boolean;
    description
        "Indicates whether PCP authentication
        is enabled/disabled";
}

leaf port-randomization-enable {
    type boolean;
    description
        "Indicates whether port randomization
        is enabled/disabled";
}

leaf port-preservation-enable {
    type boolean;
    description
        "Indicates whether port preservation
        is enabled/disabled";
}

leaf port-parity-preservation-enable {
    type boolean;
```



```
    description
      "Indicates whether port parity preservation
       is enabled/disabled";
  }

  list enabled-protocol {
    key "protocol-id";
    description
      "Indicates the set of enabled transport protocols.";

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

  leaf subscriber-mask-support{
    type boolean;
    description
      "Indicates if the subscriber-mask feature is supported";
  }

  leaf subscriber-mask {
    type uint8 {
      range "0 .. 128";
    }
    description
      "Indicates the configured subscriber-mask";
  }

  leaf port-quota {
    type uint16;
    description
      "Indicates the configured port quota.";
  }

  list exclude-ports {
    key "id";
    description
      "Indicates ports that are excluded from
       dynamic assignment.";

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



```
    uses pcplib:port-number;
}

leaf nonce-validation-checks-enable {
    type boolean;
    description
        "Indicates whether NONCE validation checks are
        enabled/disabled";
}

leaf epoch {
    type uint32;
    description
        "value of the current server's epoch.";
}

container lifetime {
    description
        "lifetime-related configuration";

    leaf minimum-lifetime {
        type uint32;
        description
            "configured minimum lifetime";
    }

    leaf maximum-lifetime {
        type uint32;
        description
            "configured maximum-lifetime";
    }
}

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

    leaf minimum-error-lifetime {
        type uint32;
        description
            "Configured minimum error lifetime,
            in seconds.";
    }

    leaf maximum-error-lifetime {
        type uint32;
        description
```



```
        "Configured maximum error lifetime,
        in seconds.";
    }
}
container mapping-table {
    description
        "Mapping table";
    list mapping-entry {
        key "index";
        description
            "mapping entry";
        uses pcp:mapping-entry;
        uses pcp:status-code;
    }
}

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 be 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 [[RFC6020](#)].

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

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

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

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

name: ietf-pcp-server
namespace: urn:ietf:params:xml:ns:yang:ietf-pcp-server
prefix: pc-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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", [BCP 81](#), [RFC 3688](#), DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", [RFC 6241](#), DOI 10.17487/RFC6241, June 2011, <<http://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, <<http://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, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc6970>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", [RFC 6991](#), DOI 10.17487/RFC6991, July 2013, <<http://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, <<http://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, <<http://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, <<http://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, <<http://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, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc7753>>.

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, <<http://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