

PCP working group
Internet-Draft
Intended status: Standards Track
Expires: August 13, 2012

D. Wing, Ed.
Cisco
S. Cheshire
Apple
M. Boucadair
France Telecom
R. Penno
Juniper Networks
P. Selkirk
ISC
February 10, 2012

Port Control Protocol (PCP)
draft-ietf-pcp-base-23

Abstract

The Port Control Protocol allows an IPv6 or IPv4 host to control how incoming IPv6 or IPv4 packets are translated and forwarded by a network address translator (NAT) or simple firewall, and also allows a host to optimize its outgoing NAT keepalive messages.

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 August 13, 2012.

Copyright Notice

Copyright (c) 2012 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	5
2.	Scope	6
2.1.	Deployment Scenarios	6
2.2.	Supported Protocols	6
2.3.	Single-homed Customer Premises Network	6
3.	Terminology	7
4.	Relationship between PCP Server and its NAT/firewall	11
5.	Note on Fixed-Size Addresses	11
6.	Common Request and Response Header Format	12
6.1.	Request Header	13
6.2.	Response Header	14
6.3.	Options	15
6.4.	Result Codes	18
7.	General PCP Operation	19
7.1.	General PCP Client: Generating a Request	20
7.2.	General PCP Server: Processing a Request	21
7.3.	General PCP Client: Processing a Response	23
7.4.	Multi-Interface Issues	24
7.5.	Epoch	24
8.	Version Negotiation	26
9.	Introduction to MAP and PEER Opcodes	27
9.1.	For Operating a Server	29
9.2.	For Operating a Symmetric Client/Server	31
9.3.	For Reducing NAT Keepalive Messages	33
9.4.	For Restoring Lost Implicit TCP Dynamic Mapping State	34
10.	MAP Opcode	35
10.1.	MAP Operation Packet Formats	36
10.2.	Generating a MAP Request	38
10.2.1.	Renewing a Mapping	39
10.3.	Processing a MAP Request	39
10.4.	Processing a MAP Response	42
10.5.	Mapping Lifetime and Deletion	42
10.6.	Address Change Events	45
10.7.	Learning the External IP Address Alone	45
11.	PEER Opcode	46
11.1.	PEER Operation Packet Formats	47
11.2.	Generating a PEER Request	50
11.3.	Processing a PEER Request	50

11.4.	Processing a PEER Response	51
12.	Options for MAP and PEER Opcodes	52
12.1.	THIRD_PARTY Option for MAP and PEER Opcodes	52
12.2.	PREFER_FAILURE Option for MAP Opcode	54
12.3.	FILTER Option for MAP Opcode	56
13.	Rapid Recovery	58
13.1.	ANNOUNCE Opcode	59
13.1.1.	ANNOUNCE operation packet format	59
13.1.2.	Generating and Processing a Solicited ANNOUNCE Message	60
13.1.3.	Generating and Processing an Unsolicited ANNOUNCE Message	60
13.2.	PCP Mapping Update	62
14.	Implementation Considerations	63
14.1.	Implementing MAP with EDM port-mapping NAT	63
14.2.	Lifetime of Explicit and Implicit Dynamic Mappings	63
14.3.	PCP Failure Scenarios	64
14.3.1.	Recreating Mappings	64
14.3.2.	Maintaining Mappings	65
14.3.3.	SCTP	65
14.4.	Source Address Replicated in PCP Header	66
14.5.	State Diagram	66
15.	Deployment Considerations	68
15.1.	Ingress Filtering	68
15.2.	Mapping Quota	69
16.	Security Considerations	69
16.1.	Simple Threat Model	69
16.1.1.	Attacks Considered	70
16.1.2.	Deployment Examples Supporting the Simple Threat Model	71
16.2.	Advanced Threat Model	71
16.3.	Residual Threats	72
16.3.1.	Denial of Service	72
16.3.2.	Ingress Filtering	73
16.3.3.	Mapping Theft	73
16.3.4.	Attacks Against Server Discovery	73
17.	IANA Considerations	73
17.1.	Port Number	73
17.2.	Opcodes	74
17.3.	Result Codes	74
17.4.	Options	74
18.	Acknowledgments	74
19.	References	75
19.1.	Normative References	75
19.2.	Informative References	76
Appendix A.	NAT-PMP Transition	78
Appendix B.	Change History	79
B.1.	Changes from draft-ietf-pcp-base-22 to -23	79

B.2.	Changes from draft-ietf-pcp-base-21 to -22	80
B.3.	Changes from draft-ietf-pcp-base-20 to -21	80
B.4.	Changes from draft-ietf-pcp-base-19 to -20	81
B.5.	Changes from draft-ietf-pcp-base-18 to -19	81
B.6.	Changes from draft-ietf-pcp-base-17 to -18	81
B.7.	Changes from draft-ietf-pcp-base-16 to -17	82
B.8.	Changes from draft-ietf-pcp-base-15 to -16	82
B.9.	Changes from draft-ietf-pcp-base-14 to -15	82
B.10.	Changes from draft-ietf-pcp-base-13 to -14	83
B.11.	Changes from draft-ietf-pcp-base-12 to -13	83
B.12.	Changes from draft-ietf-pcp-base-11 to -12	84
B.13.	Changes from draft-ietf-pcp-base-10 to -11	84
B.14.	Changes from draft-ietf-pcp-base-09 to -10	84
B.15.	Changes from draft-ietf-pcp-base-08 to -09	84
B.16.	Changes from draft-ietf-pcp-base-07 to -08	86
B.17.	Changes from draft-ietf-pcp-base-06 to -07	87
B.18.	Changes from draft-ietf-pcp-base-05 to -06	88
B.19.	Changes from draft-ietf-pcp-base-04 to -05	89
B.20.	Changes from draft-ietf-pcp-base-03 to -04	90
B.21.	Changes from draft-ietf-pcp-base-02 to -03	90
B.22.	Changes from draft-ietf-pcp-base-01 to -02	91
B.23.	Changes from draft-ietf-pcp-base-00 to -01	91
Authors' Addresses		92

1. Introduction

The Port Control Protocol (PCP) provides a mechanism to control how incoming packets are forwarded by upstream devices such as NAT64, NAT44, IPv6 and IPv4 firewall devices, and a mechanism to reduce application keepalive traffic. PCP is designed to be implemented in the context of Carrier-Grade NATs (CGNs), small NATs (e.g., residential NATs), as well as with dual-stack and IPv6-only CPE routers, and all of the currently-known transition scenarios towards IPv6-only CPE routers. PCP allows hosts to operate servers for a long time (e.g., a webcam) or a short time (e.g., while playing a game or on a phone call) when behind a NAT device, including when behind a CGN operated by their Internet service provider or an IPv6 firewall integrated in their CPE router.

PCP allows applications to create mappings from an external IP address and port to an internal IP address and port. These mappings are required for successful inbound communications destined to machines located behind a NAT or a firewall.

After creating a mapping for incoming connections, it is necessary to inform remote computers about the IP address and port for the incoming connection. This is usually done in an application-specific manner. For example, a computer game might use a rendezvous server specific to that game (or specific to that game developer), a SIP phone would use a SIP proxy, and a client using DNS-Based Service Discovery [[I-D.cheshire-dnsext-dns-sd](#)] would use DNS Update [[RFC2136](#)] [[RFC3007](#)]. PCP does not provide this rendezvous function. The rendezvous function may support IPv4, IPv6, or both. Depending on that support and the application's support of IPv4 or IPv6, the PCP client may need an IPv4 mapping, an IPv6 mapping, or both.

Many NAT-friendly applications send frequent application-level messages to ensure their session will not be timed out by a NAT. These are commonly called "NAT keepalive" messages, even though they are not sent to the NAT itself (rather, they are sent 'through' the NAT). These applications can reduce the frequency of such NAT keepalive messages by using PCP to learn (and influence) the NAT mapping lifetime. This helps reduce bandwidth on the subscriber's access network, traffic to the server, and battery consumption on mobile devices.

Many NATs and firewalls include application layer gateways (ALGs) to create mappings for applications that establish additional streams or accept incoming connections. ALGs incorporated into NATs may also modify the application payload. Industry experience has shown that these ALGs are detrimental to protocol evolution. PCP allows an application to create its own mappings in NATs and firewalls,

reducing the incentive to deploy ALGs in NATs and firewalls.

2. Scope

2.1. Deployment Scenarios

PCP can be used in various deployment scenarios, including:

- o Basic NAT [[RFC3022](#)]
- o Network Address and Port Translation [[RFC3022](#)], such as commonly deployed in residential NAT devices
- o Carrier-Grade NAT [[I-D.ietf-behave-lsn-requirements](#)]
- o Dual-Stack Lite (DS-Lite) [[RFC6333](#)]
- o Layer-2 Aware NAT [[I-D.miles-behave-l2nat](#)]
- o Dual-Stack Extra Lite [[I-D.arkko-dual-stack-extra-lite](#)]
- o NAT64, both Stateless [[RFC6145](#)] and Stateful [[RFC6146](#)]
- o IPv4 and IPv6 simple firewall control [[RFC6092](#)]
- o IPv6-to-IPv6 Network Prefix Translation (NPTv6) [[RFC6296](#)]

2.2. Supported Protocols

The PCP Opcodes defined in this document are designed to support transport-layer protocols that use a 16-bit port number (e.g., TCP, UDP, SCTP [[RFC4960](#)], DCCP [[RFC4340](#)]). Protocols that do not use a port number (e.g., RSVP, IPsec ESP, ICMP, ICMPv6) are supported for IPv4 firewall, IPv6 firewall, and NPTv6 functions, but are out of scope for any NAT functions.

2.3. Single-homed Customer Premises Network

PCP assumes a single-homed IP address model. That is, for a given IP address of a host, only one default route exists to reach the Internet from that source IP address. This is important because after a PCP mapping is created and an inbound packet (e.g., TCP SYN) arrives at the host, the outbound response (e.g., TCP SYNACK) has to go through the same path so it is seen by the firewall or rewritten by the NAT. This restriction exists because otherwise there would need to be a PCP-enabled NAT for every egress (because the host could not reliably determine which egress path packets would take) and the

client would need to be able to reliably make the same internal/external mapping in every NAT gateway, which in general is not possible (because the other NATs might have the necessary port mapped to another host).

3. Terminology

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 "Key words for use in RFCs to Indicate Requirement Levels" [[RFC2119](#)].

Internal Host:

A host served by a NAT gateway, or protected by a firewall. This is the host that receives the incoming traffic resulting from a PCP MAP request, or the host that initiated an implicit dynamic outbound mapping (e.g., by sending a TCP SYN) across a firewall or a NAT.

Remote Peer Host:

A host with which an Internal Host is communicating. This can include another Internal Host (or even the same Internal Host); if a NAT is involved, the NAT would need to hairpin the traffic.

Internal Address:

The address of an Internal Host served by a NAT gateway or protected by a firewall.

External Address:

The address of an Internal Host as seen by other Remote Peers on the Internet with which the Internal Host is communicating, after translation by any NAT gateways on the path. An External Address is generally a public routable (i.e., non-private) address. In the case of an Internal Host protected by a pure firewall, with no address translation on the path, its External Address is the same as its Internal Address.

Endpoint-Dependent Mapping (EDM): A term applied to NAT operation where an implicit mapping created by outgoing traffic (e.g., TCP SYN) from a single Internal Address and Port to different Remote Peers and Ports may be assigned different External Ports, and a subsequent PCP MAP request for that Internal Address and Port may be assigned yet another different External Port. This term encompasses both Address-Dependent Mapping and Address and Port-Dependent Mapping [[RFC4787](#)].

Remote Peer Address:

The address of a Remote Peer, as seen by the Internal Host. A Remote Address is generally a publicly routable address. In the case of a Remote Peer that is itself served by a NAT gateway, the Remote Address may in fact be the Remote Peer's External Address, but since this remote translation is generally invisible to software running on the Internal Host, the distinction can safely be ignored for the purposes of this document.

Third Party:

In the common case, an Internal Host manages its own Mappings using PCP requests, and the Internal Address of those Mappings is the same as the source IP address of the PCP request packet.

In the case where one device is managing Mappings on behalf of some other device that does not implement PCP, the presence of the THIRD_PARTY Option in the MAP request signifies that the specified address, rather than the source IP address of the PCP request packet, should be used as the Internal Address for the Mapping.

Mapping, Port Mapping, Port Forwarding:

A NAT mapping creates a relationship between an internal IP address, protocol, and port, and an external IP address, protocol, and port. More specifically, it creates a translation rule where packets destined to the external IP and port are translated to the internal IP and port, and vice versa. In the case of a pure firewall, the "Mapping" is the identity function, translating an internal IP address and port number to the same external IP address and port number. Firewall filtering, applied in addition to that identity mapping function, is separate from the mapping itself.

Mapping Types:

There are two dimensions to classifying mapping types: how they are created (implicitly/explicitly) and their primary purpose (outbound/inbound).

- * Implicit dynamic mappings are created implicitly as a side-effect of traffic such as an outgoing TCP SYN or outgoing UDP packet. Such packets were not originally designed explicitly for creating NAT (or firewall) state, but they can have that effect when they pass through a NAT (or firewall) device.
- * Explicit dynamic mappings are created as a result of explicit PCP MAP and PEER requests. Like a DHCP address lease, explicit dynamic mappings have finite lifetime, and, as with a DHCP address lease, if a client wants a mapping to persist the client must prove that it is still present by periodically

renewing the mapping to prevent it from expiring. If a PCP client goes away, then any mappings it created will be automatically cleaned up when they expire.

- * Explicit static mappings are created by manual configuration (e.g., via command-line interface or web-based user interface) and persist until the user changes that manual configuration.

Both implicit and explicit dynamic mappings are dynamic in the sense that they are created on demand, as requested (implicitly or explicitly) by the Internal Host, and have a lifetime. After the lifetime, the mapping is deleted unless the lifetime is extended by action by the Internal Host (e.g., sending more traffic or sending a new PCP request).

Explicit static mappings differ from explicit dynamic mappings in that their lifetime is effectively infinite (they exist until manually removed) but otherwise they behave exactly the same as explicit MAP mappings.

While all mappings are by necessity bidirectional (most Internet communication requires information to flow in both directions for successful operation) it can be helpful when talking about mappings to identify them loosely according to their 'primary' purpose.

- * Outbound mappings exist primarily to enable outbound communication. For example, when a client calls `connect()` to make an outbound connection, a NAT gateway will create an implicit dynamic outbound mapping to facilitate that outbound communication.
- * Inbound mappings exist primarily to enable listening servers to receive inbound connections. Generally, when a client calls `listen()` to listen for inbound connections, a NAT gateway will not implicitly create any mapping to facilitate that inbound communication. A PCP MAP request can be used explicitly to create a dynamic inbound mapping to enable the desired inbound communication.

Explicit static (manual) mappings and explicit dynamic (MAP) mappings both allow Internal Hosts to receive inbound traffic that is not in direct response to any immediately preceding outbound communication (i.e., to allow Internal Hosts to operate a "server" that is accessible to other hosts on the Internet).

PCP Client:

A PCP software instance responsible for issuing PCP requests to a PCP server. Several independent PCP Clients can exist on the same host (just as several independent web browsers can exist on the same host). Several PCP Clients can be located in the same local network. A PCP Client can issue PCP requests on behalf of a third party device for which it is authorized to do so. An interworking function from Universal Plug and Play Internet Gateway Device (UPnP IGDv1 [[IGDv1](#)]) to PCP is another example of a PCP Client. A PCP server in a NAT gateway that is itself a client of another NAT gateway (nested NAT) may itself act as a PCP client to the upstream NAT.

PCP-Controlled Device:

A NAT or firewall that controls or rewrites packet flows between internal hosts and remote peer hosts. PCP manages the Mappings on this device.

PCP Server:

A PCP software instance that implements the server side of the PCP protocol, via which PCP clients request and manage explicit mappings. See also [Section 4](#).

Interworking Function:

A functional element responsible for translating or proxying another protocol to PCP. For example interworking UPnP IGDv1 [[IGDv1](#)] with PCP.

Subscriber:

The unit of billing for a commercial ISP. A subscriber may have a single IP address from the commercial ISP (which can be shared among multiple hosts using a NAT gateway, thereby making them appear to be a single host to the ISP) or may have multiple IP addresses provided by the commercial ISP. In either case, the IP address or addresses provided by the ISP may themselves be further translated by a large-scale NAT operated by the ISP.

4. Relationship between PCP Server and its NAT/firewall

The PCP server receives and responds to PCP requests. The PCP server functionality is typically a capability of a NAT or firewall device, as shown in Figure 1. It is also possible for the PCP functionality to be provided by some other device, which communicates with the actual NAT or firewall via some other proprietary mechanism, as long as from the PCP client's perspective such split operation is indistinguishable from the integrated case.

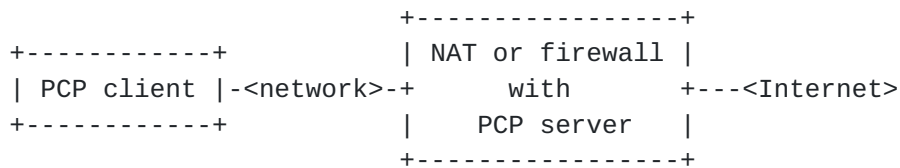


Figure 1: PCP-Enabled NAT or Firewall

A NAT or firewall device, between the PCP client and the Internet, might implement simple or advanced firewall functionality. This may be a side-effect of the technology implemented by the device (e.g., a network address and port translator, by virtue of its port rewriting, normally requires connections to be initiated from an inside host towards the Internet), or this might be an explicit firewall policy to deny unsolicited traffic from the Internet. Some firewall devices deny certain unsolicited traffic from the Internet (e.g., TCP, UDP to most ports) but allow certain other unsolicited traffic from the Internet (e.g., UDP port 500 and IPsec ESP) [[RFC6092](#)]. Such default filtering (or lack thereof) is out of scope of PCP itself. If a device supports PCP and wants to receive traffic, and does not possess knowledge of such filtering, it SHOULD use PCP to create the necessary mappings to receive the desired traffic.

5. Note on Fixed-Size Addresses

For simplicity in building and parsing request and response packets, PCP always uses fixed-size 128-bit IP address fields for both IPv6 addresses and IPv4 addresses.

When the address field holds an IPv6 address, the fixed-size 128-bit IP address field holds the IPv6 address stored as-is.

When the address field holds an IPv4 address, IPv4-mapped IPv6 addresses [[RFC4291](#)] are used (::ffff:0:0/96). This has the first 80 bits set to zero and the next 16 set to one, while its last 32 bits are filled with the IPv4 address. This is unambiguously distinguishable from a native IPv6 address, because IPv4-mapped IPv6

address [[RFC4291](#)] are not used as either the source or destination address of actual IPv6 packets.

When checking for an IPv4-mapped IPv6 address, all of the first 96 bits MUST be checked for the pattern -- it is not sufficient to check for ones in bits 81-96.

The all-zeroes IPv6 address is expressed by filling the fixed-size 128-bit IP address field with all zeroes (::).

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

6. Common Request and Response Header Format

All PCP messages contain a request (or response) header containing an Opcode, any relevant Opcode-specific information, and zero or more Options. The packet layout for the common header, and operation of the PCP client and PCP server, are described in the following sections. The information in this section applies to all Opcodes. Behavior of the Opcodes defined in this document is described in [Section 10](#) and [Section 11](#).

6.1. Request Header

All requests have the following format:

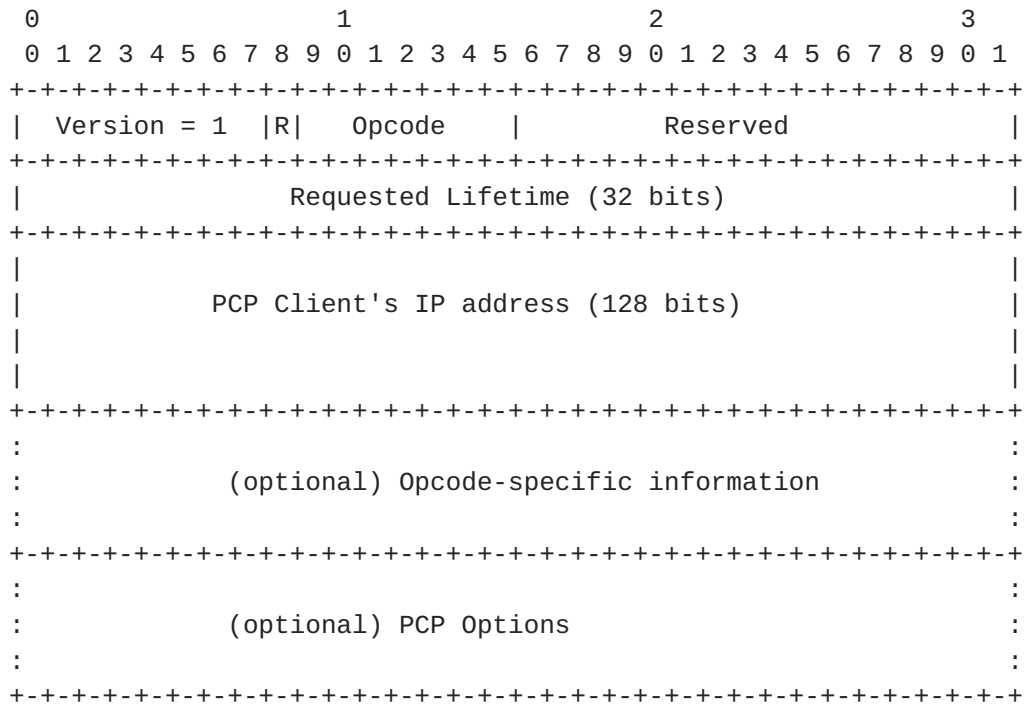


Figure 2: Common Request Packet Format

These fields are described below:

Version: This document specifies protocol version 1. This value MUST be 1 when sending, and MUST be 1 when receiving. This field is used for version negotiation as described in [Section 8](#).

R: Indicates Request (0) or Response (1). All Requests MUST use 0.

Opcode: A seven-bit value specifying the operation to be performed. Opcodes are defined in [Section 10](#) and [Section 11](#).

Reserved: 16 reserved bits. MUST be 0 on transmission and MUST be ignored on reception.

Requested Lifetime: An unsigned 32-bit integer, in seconds, ranging from 0 to $2^{32}-1$ seconds. This is used by the MAP and PEER Opcodes defined in this document for their requested lifetime. Future Opcodes which don't need this field MUST set the field to zero on transmission and ignored on reception.

PCP Client's IP Address: The source IPv4 or IPv6 address in the IP header used by the PCP client when sending this PCP request. IPv4 is represented using an IPv4-mapped IPv6 address.

6.2. Response Header

All responses have the following format:

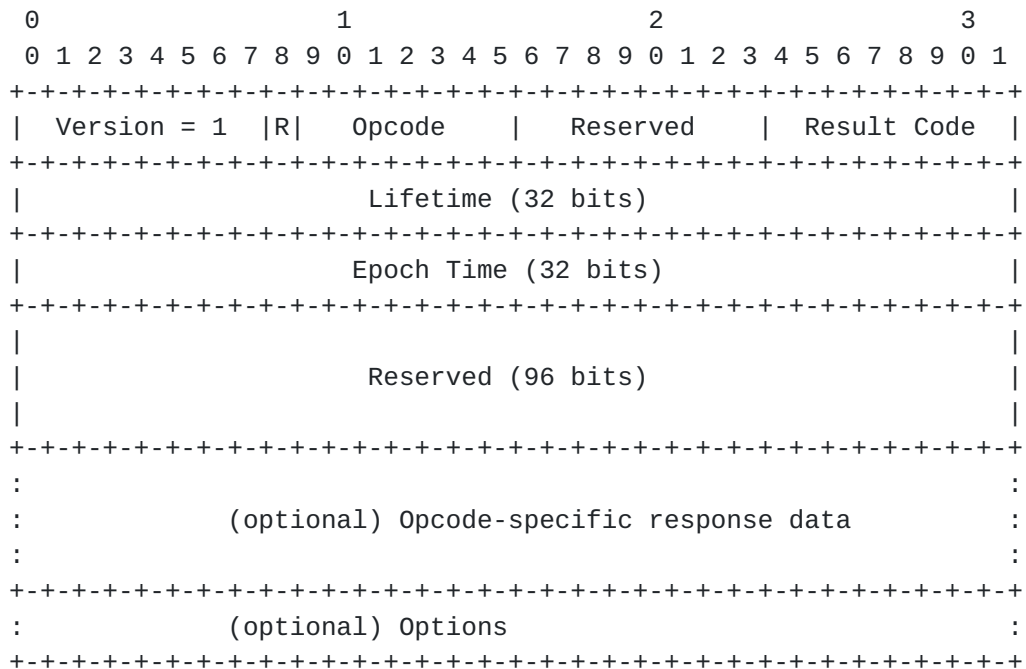


Figure 3: Common Response Packet Format

These fields are described below:

Version: Responses MUST use version 1. This is set by the server.

R: Indicates Request (0) or Response (1). All Responses MUST use 1. This is set by the server.

Opcode: The 7-bit Opcode value. The server copies this value from the request.

Reserved: 8 reserved bits, MUST be sent as 0, MUST be ignored when received. This is set by the server.

Result Code: The result code for this response. See [Section 6.4](#) for values. This is set by the server.

Lifetime: An unsigned 32-bit integer, in seconds, ranging from 0 to $2^{32}-1$ seconds. On an error response, this indicates how long clients should assume they'll get the same error response from that PCP server if they repeat the same request. On a success response for the PCP Opcodes that create a mapping (MAP and PEER), the Lifetime field indicates the lifetime for this mapping. If future Opcodes are defined that do not have a lifetime associated with them, then in success responses for those Opcodes the Lifetime MUST be set to zero on transmission and MUST be ignored on reception. This is set by the server.

Epoch Time: The server's Epoch time value. See [Section 7.5](#) for discussion. This value is set by the server, in both success and error responses.

Reserved: 96 reserved bits. For requests that were successfully parsed, this MUST be sent as 0, MUST be ignored when received. This is set by the server. For requests that were not successfully parsed, the server blindly copies the 96 bits from the request message into a response.

6.3. Options

A PCP Opcode can be extended with one or more Options. Options can be used in requests and responses. The design decisions in this specification about whether to include a given piece of information in the base Opcode format or in an Option were an engineering trade-off between packet size and code complexity. For information that is usually (or always) required, placing it in the fixed Opcode data results in simpler code to generate and parse the packet, because the information is a fixed location in the Opcode data, but wastes space in the packet in the event that field is all-zeroes because the information is not needed or not relevant. For information that is required less often, placing it in an Option results in slightly more complicated code to generate and parse packets containing that Option, but saves space in the packet when that information is not needed. Placing information in an Option also means that an implementation that never uses that information doesn't even need to implement code to generate and parse it. For example, a client that never requests mappings on behalf of some other device doesn't need to implement code to generate the THIRD_PARTY Option, and a PCP server that doesn't implement the necessary security measures to create third-party mappings safely doesn't need to implement code to parse the THIRD_PARTY Option.

If, while processing an Option, an error is encountered that causes a PCP error response to be generated, the PCP request MUST cause no state change in the PCP server or the PCP-controlled device (i.e., it rolls back any changes it might have made while processing the request). The response MUST consist of a complete copy of the request packet with the error code and other appropriate fields set in the header. An Option MAY appear more than once in a request or in a response, if permitted by the definition of the Option. If the Option's definition allows the Option to appear only once but it

appears more than once in a request, and the Option is understood by the PCP server, the PCP server **MUST** respond with the **MALFORMED_OPTION** result code; if this occurs in a response, the PCP client processes the first occurrence and **MAY** log an error. If the PCP server encounters an invalid option (e.g., option length extends beyond the length of the PCP Opcode itself), the error **MALFORMED_OPTION** **SHOULD** be returned (rather than **MALFORMED_REQUEST**), as that helps the client better understand how the packet was malformed. If a PCP response would have exceeded the maximum PCP message size, the PCP server **SHOULD** respond with **MALFORMED_REQUEST**.

If an Option cannot successfully be parsed, the PCP server **MUST** generate an error response with code **MALFORMED_OPTION**. The most significant bit in the Option Code indicates if its processing is optional or mandatory. If the most significant bit is set, handling this Option is optional, and a PCP server **MAY** process or ignore this Option, entirely at its discretion. If the most significant bit is clear, handling this Option is mandatory, and a PCP server **MUST** return the error **UNSUPP_OPTION** if the Option is unrecognized, unimplemented, or disabled, or if the client is not authorized to use the Option. All processed options are included in successful responses, but unprocessed options are not included in successful responses. All options are returned in error responses.

PCP clients are free to ignore any or all Options included in responses, although naturally if a client explicitly requests an Option where correct execution of that Option requires processing the Option data in the response, that client is expected to implement code to do that.

Different options are valid for different Opcodes. For example:

- o The **THIRD_PARTY** Option is valid for both **MAP** and **PEER** Opcodes.
- o The **FILTER** Option is valid only for the **MAP** Opcode (for the **PEER** Opcode it would have no meaning).
- o The **PREFER_FAILURE** Option is valid only for the **MAP** Opcode (for the **PEER** Opcode, similar semantics are automatically implied).

Option definitions MUST include the information below:

Option Name: <mnemonic>
Number: <value>
Purpose: <textual description>
Valid for Opcodes: <list of Opcodes>
Length: <rules for length>
May appear in: <requests/responses/both>
Maximum occurrences: <count>

6.4. Result Codes

The following result codes may be returned as a result of any Opcode received by the PCP server. The only success result code is 0; other values indicate an error. If a PCP server encounters multiple errors during processing of a request, it SHOULD use the most specific error message. Each error code below is classified as either a 'long lifetime' error or a 'short lifetime' error, which provides guidance to PCP server developers for the value of the Lifetime field for these errors. It is RECOMMENDED that short lifetime errors use a 30 second lifetime and long lifetime errors use a 30 minute lifetime.

- 0 SUCCESS: Success.
- 1 UNSUPP_VERSION: Unsupported protocol version.
- 2 NOT_AUTHORIZED: 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.
- 3 MALFORMED_REQUEST: The request could not be successfully parsed.
- 4 UNSUPP_OPCODE: Unsupported Opcode.
- 5 UNSUPP_OPTION: Unsupported Option. This error only occurs if the Option is in the mandatory-to-process range.
- 6 MALFORMED_OPTION: Malformed Option (e.g., appears too many times, invalid length).
- 7 NETWORK_FAILURE: The PCP server or the device it controls are experiencing a network failure of some sort (e.g., has not obtained an External IP address). This is a short lifetime error.

- 8 NO_RESOURCES: 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 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 is a short lifetime error. This can be used as a catch-all error, should no other error message be suitable.
- 9 UNSUPP_PROTOCOL: Unsupported Protocol. This is a long lifetime error.
- 10 USER_EX_QUOTA: Mapping would exceed user's port quota. This is a short lifetime error.
- 11 CANNOT_PROVIDE_EXTERNAL: the suggested external port and/or external address cannot be provided. This error MUST only be returned for PEER requests, for MAP requests that included the PREFER_FAILURE Option (because otherwise a new external port could have been assigned), or MAP requests for the SCTP protocol. See [Section 12.2](#) for processing details. The error lifetime depends on the reason for the failure.
- 12 ADDRESS_MISMATCH: the source IP address of the request packet does not match the contents of the PCP Client's IP Address field.
- 13 EXCESSIVE_REMOTE_PEERS: 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. See [Section 12.3](#) for processing information. This is a long lifetime error.

7. General PCP Operation

PCP messages MUST be sent over UDP [[RFC0768](#)]. Every PCP request generates at least one response, so PCP does not need to run over a reliable transport protocol.

PCP is idempotent, meaning that if the PCP client sends the same request multiple times (or the PCP client sends the request once and it is duplicated by the network), and the PCP server processes those requests multiple times, the result is the same as if the PCP server had processed only one of those duplicate requests. Note this doesn't mean that all responses are the same if there was a state change in the PCP-controlled device. For example, if a request is received while the PCP-controlled device has no mappings available, it will generate an error response. If mappings become available and then a (duplicated or re-transmitted) request is seen by the server,

it will generate a non-error response. Of course, this sort of behavior is common to all UDP-based protocols and is not unique to PCP. A PCP client will need to properly handle such updated responses for any request it sends, most notably to support Mapping Update ([Section 13.2](#)).

7.1. General PCP Client: Generating a Request

This section details operation specific to a PCP client, for any Opcode. Procedures specific to the MAP Opcode are described in [Section 10](#), and procedures specific to the PEER Opcode are described in [Section 11](#).

Prior to sending its first PCP message, the PCP client determines which server to use. The PCP client performs the following steps to determine its PCP server:

1. if a PCP server is configured (e.g., in a configuration file or via DHCP), that single configuration source is used as the list of PCP Server(s), else;
2. the default router list (for IPv4 and IPv6) is used as the list of PCP Server(s).

For the purposes of this document, only a single PCP server address is supported. Should future specifications define configuration methods that provide a list of PCP server addresses, those specifications will define how clients select one or more addresses from that list.

With that PCP server address, the PCP client formulates its PCP request. The PCP request contains a PCP common header, PCP Opcode and payload, and (possibly) Options. As with all UDP or TCP client software on any operating system, when several independent PCP clients exist on the same host, each uses a distinct source port number to disambiguate their requests and replies. The PCP client's source port SHOULD be randomly generated [[RFC6056](#)].

To assist with detecting an on-path NAT, the PCP client MUST include the source IP address of the PCP message in the PCP request. This is typically its own IP address; see [Section 14.4](#) for how this can be coded.

When attempting to contact a PCP server, the PCP client sends a PCP message to the first server in its list of PCP servers, and SHOULD initialize a timer to a random value between 2 and 3 seconds.

For as long as the client still desires the indicated mapping, if no

response is received before the timer expires then the request is re-transmitted. These re-transmissions are sent at a random value between 1.5 and 2.5 of the previous transmission interval. The randomization helps avoid client synchronization. This repeats until a maximum retransmission interval between 512 and 1024 seconds, at which point the retransmission continue at that interval, until a successful response is received or the client no longer desires the indicated mapping.

Once a PCP client has successfully received a response from a PCP server on that interface, it sends subsequent PCP requests to that same server, with a retransmission timer of 2 seconds. If, after 2 seconds, a response is not received from that PCP server, the same back-off algorithm described above is performed.

7.2. General PCP Server: Processing a Request

This section details operation specific to a PCP server. Processing SHOULD be performed in the order of the following paragraphs.

A PCP server MUST only accept normal (non-THIRD_PARTY) PCP requests from a client on the same interface it would normally receive packets from that client, and MUST silently ignore PCP requests arriving on any other interface. For example, a residential NAT gateway accepts PCP requests only when they arrive on its (LAN) interface connecting to the internal network, and silently ignores any PCP requests arriving on its external (WAN) interface. A PCP server which supports THIRD_PARTY requests MAY be configured to accept THIRD_PARTY requests on other interfaces from properly authorized clients.

Upon receiving a request, the PCP server parses and validates it. A valid request contains a valid PCP common header, one valid PCP Opcode, and zero or more Options (which the server might or might not comprehend). If an error is encountered during processing, the server generates an error response which is sent back to the PCP client. Processing an Opcode and the Options are specific to each Opcode.

Error responses have the same packet layout as success responses, with certain fields from the request copied into the response, and other fields assigned by the PCP server set as indicated in Figure 3.

Copying request fields into the response is important because this is what enables a client to identify to which request a given response pertains. For Opcodes that are understood by the PCP server, it follows the requirements of that Opcode to copy the appropriate fields. For Opcodes that are not understood by the PCP server (including the ANNOUNCE Opcode), it simply generates the

UNSUPP_OPCODE response and copies fields from the PCP header and copies the rest of the PCP payload as-is (without attempting to interpret it).

All responses (both error and success) contain the same Opcode as the request, but with the "R" bit set.

Any error response has a nonzero Result Code, and is created by:

- o Copying the entire request packet, or 1024 octets, whichever is less, and zero-padding the response to a multiple of 4 octets if necessary
- o Setting the R bit
- o Setting the Result Code
- o Setting the Lifetime, Epoch Time and Reserved fields
- o Possibly updating other fields in the response if appropriate

A success response has a zero Result Code, and is created by:

- o Copying the first four octets of request packet header
- o Setting the R bit
- o Setting the Result Code to zero
- o Setting the Lifetime, Epoch Time and Reserved fields
- o Possibly setting opcode-specific response data if appropriate
- o Adding any processed options to the response message

If the received PCP request message is less than two octets long it is silently dropped.

If the R bit is set the message is silently dropped.

If the first octet (version) is a version that is not supported, a response is generated with the UNSUPP_VERSION result code, and the other steps detailed in [Section 8](#) are followed.

Otherwise, if the version is supported but the received message is shorter than 24 octets, the message is silently dropped.

If the server is overloaded by requests (from a particular client or from all clients), it MAY simply silently discard requests, as the requests will be retried by PCP clients, or it MAY generate the NO_RESOURCES error response.

If the length of the message exceeds 1024 octets, is not a multiple of 4 octets, or is too short for the opcode in question, it is invalid and a MALFORMED_REQUEST response is generated.

The PCP server compares the source IP address (from the received IP header) with the field PCP Client IP Address. If they do not match, the error ADDRESS_MISMATCH MUST be returned. This is done to detect

and prevent accidental use of PCP where a non-PCP-aware NAT exists between the PCP client and PCP server. If the PCP client wants such a mapping it needs to ensure the PCP field matches its apparent IP address from the perspective of the PCP server.

7.3. General PCP Client: Processing a Response

The PCP client receives the response and verifies that the source IP address and port belong to the PCP server of a previously-sent PCP request. If not, the response is silently dropped.

If the received PCP response message is less than four octets long it is silently dropped.

If the R bit is clear the message is silently dropped.

If the error code is UNSUPP_VERSION processing continues as described in [Section 8](#).

The PCP client then validates that the Opcode matches a previous PCP request. Responses shorter than 24 octets, longer than 1024 octets, or not a multiple of 4 octets are invalid and ignored, likely causing the request to be re-transmitted. The response is further matched by comparing fields in the response Opcode-specific data to fields in the request Opcode-specific data, as described by the processing for that Opcode. After these matches are successful, the PCP client checks the Epoch Time field to determine if it needs to restore its state to the PCP server (see [Section 7.5](#)). A PCP client SHOULD be prepared to receive multiple responses from the PCP Server at any time after a single request is sent. This allows the PCP server to inform the client of mapping changes such as an update or deletion. For example, a PCP Server might send a SUCCESS response and, after a configuration change on the PCP Server, later send a NOT_AUTHORIZED response.

If the error ADDRESS_MISMATCH is received, it indicates the presence of a NAT between the PCP client and PCP server. Procedures to resolve this problem are beyond the scope of this document.

For both success and error responses a Lifetime value is returned. The Lifetime indicates how long this request is considered valid by the server. The PCP client SHOULD impose an upper limit on this returned value (to protect against absurdly large values, e.g., 5 years). A limit of 24 hours for success responses and 30 minutes for error responses is RECOMMENDED.

If the result code is 0 (SUCCESS), the request succeeded.

If the result code is not 0, the request failed. If the result code is NO_RESOURCES, the PCP client SHOULD NOT send further requests for new mappings to that PCP server for the (limited) value of the Lifetime. If a request for renewal or deletion of an existing mapping results in NO_RESOURCES, the PCP client SHOULD NOT send further requests of any kind to that PCP server for the (limited) value of the Lifetime. For other error result codes, the PCP client SHOULD NOT resend the same request for the (limited) value of the Lifetime.

If the PCP client has discovered a new PCP server (e.g., connected to a new network), the PCP client MAY immediately begin communicating with this PCP server, without regard to hold times from communicating with a previous PCP server.

7.4. Multi-Interface Issues

Hosts which desire a PCP mapping might be multi-interfaced (i.e., own several logical/physical interfaces). Indeed, a host can be configured with several IPv4 addresses (e.g., WiFi and Ethernet) or dual-stacked. These IP addresses may have distinct reachability scopes (e.g., if IPv6 they might have global reachability scope as for Global Unicast Address (GUA, [RFC3587](#)) or limited scope as for Unique Local Address (ULA) [RFC4193](#))).

IPv6 addresses with global reachability (e.g., GUA) SHOULD be used as the source address when generating a PCP request. IPv6 addresses without global reachability (e.g., ULA [RFC4193](#)), SHOULD NOT be used as the source interface when generating a PCP request. If IPv6 privacy addresses [RFC4941](#) are used for PCP mappings, a new PCP request will need to be issued whenever the IPv6 privacy address is changed. This PCP request SHOULD be sent from the IPv6 privacy address itself. It is RECOMMENDED that mappings to the previous privacy address be deleted.

Due to the ubiquity of IPv4 NAT, IPv4 addresses with limited scope (e.g., private addresses [RFC1918](#)) MAY be used as the source interface when generating a PCP request.

7.5. Epoch

Every PCP response sent by the PCP server includes an Epoch time field. This time field increments by one every second. Anomalies in the received Epoch time value provide a hint to PCP clients that a PCP server state loss may have occurred. Clients respond to such state loss hints by promptly renewing their mappings, so as to quickly restore any lost state at the PCP server.

If the PCP server resets or loses the state of its explicit dynamic Mappings (that is, those mappings created by PCP requests), due to reboot, power failure, or any other reason, it MUST reset its Epoch time to its initial starting value (usually zero) to provide this hint to PCP clients. After resetting its Epoch time, the PCP server resumes incrementing the Epoch time value by one every second. Similarly, if the public IP address(es) of the NAT (controlled by the PCP server) changes, the Epoch time MUST be reset. A PCP server MAY maintain one Epoch time value for all PCP clients, or MAY maintain distinct Epoch time values (per PCP client, per interface, or based on other criteria); this choice is implementation-dependent.

Whenever a client receives a PCP response, the client validates the received Epoch time value according to the procedure below, using integer arithmetic:

- o If this is the first PCP response the client has received from this PCP server, the Epoch time value is treated as necessarily valid, otherwise
 - * If the current PCP server Epoch time value (`current_server_time`) is less than the previously received PCP server Epoch time value (`previous_server_time`) then the client treats the Epoch time value as obviously invalid (time should not go backwards), else
 - + The client computes the difference between its current local time value (`current_client_time`) and the time the previous PCP response was received from this PCP server (`previous_client_time`):
`client_delta = current_client_time - previous_client_time;`
 - + The client computes the difference between the current PCP server Epoch time value (`current_server_time`) and the previously received Epoch time value (`previous_server_time`):
`server_delta = current_server_time - previous_server_time;`
 - + If `client_delta+2 < server_delta - server_delta/16`
or `server_delta+2 < client_delta - client_delta/16`
then the client treats the Epoch time value as invalid,
else the client treats the Epoch time value as valid
- o The client records the current time values for use in its next comparison:
`previous_client_time = current_client_time`
`previous_server_time = current_server_time`

If the PCP client determined that the Epoch time value it received was invalid then it concludes that the PCP server may have lost state, and promptly renews all its active port mapping leases as described in [Section 14.3.1](#).

Notes:

- o The client clock MUST never go backwards. If `current_client_time` is found to be less than `previous_client_time` then this is a client bug, and how the client deals with this client bug is implementation specific.
- o The calculations above are constructed to allow `client_delta` and `server_delta` to be computed as unsigned integer values.
- o The "+2" in the calculations above is to accommodate quantization errors in client and server clocks (up to one second quantization error each in server and client time intervals).
- o The "/16" in the calculations above is to accommodate inaccurate clocks in low-cost devices. This allows for a total discrepancy of up to 1/16 (6.25%) to be considered benign, e.g., if one clock were to run too fast by 3% while the other clock ran too slow by 3% then the client would not consider this difference to be anomalous or indicative of a restart having occurred. This tolerance is strict enough to be effective at detecting reboots, while not being so strict as to generate false alarms.

8. Version Negotiation

A PCP client sends its requests using PCP version number 1. Should later updates to this document specify different message formats with a version number greater than 1 it is expected that PCP servers will still support version 1 in addition to the newer version(s). However, in the event that a server returns a response with result code UNSUPP_VERSION, the client MAY log an error message to inform the user that it is too old to work with this server.

Should later updates to this document specify different message formats with a version number greater than 1, and backwards compatibility is desired, this first octet can be used for forward and backward compatibility.

If future PCP versions greater than 1 are specified, version negotiation proceeds as follows:

1. If a client or server supports more than one version it SHOULD support a contiguous range of versions -- i.e., a lowest version and a highest version and all versions in between.
2. The client sends its first request using the highest (i.e., presumably 'best') version number it supports.
3. If the server supports that version it responds normally.
4. If the server does not support that version it replies giving a result containing the result code UNSUPP_VERSION, and the closest version number it does support (if the server supports a range of versions higher than the client's requested version, the server returns the lowest of that supported range; if the server supports a range of versions lower than the client's requested version, the server returns the highest of that supported range).
5. If the client receives an UNSUPP_VERSION result containing a version it does support, it records this fact and proceeds to use this message version for subsequent communication with this PCP server (until a possible future UNSUPP_VERSION response if the server is later updated, at which point the version negotiation process repeats).
6. If the client receives an UNSUPP_VERSION result containing a version it does not support then the client MAY log an error message to inform the user that it is too old to work with this server, and the client SHOULD set a timer to retry its request in 30 minutes or the returned Lifetime value, whichever is smaller.

9. Introduction to MAP and PEER Opcodes

There are four uses for the MAP and PEER Opcodes defined in this document:

- o a host operating a server and wanting an incoming connection ([Section 9.1](#));
- o a host operating a client and server on the same port ([Section 9.2](#));
- o a host operating a client and wanting to optimize the application keepalive traffic ([Section 9.3](#));
- o and a host operating a client and wanting to restore lost state in its NAT ([Section 9.4](#)).

These are discussed in the following sections, and a (non-normative) state diagram is provided in [Section 14.5](#).

When operating a server ([Section 9.1](#) and [Section 9.2](#)) the PCP client knows if it wants an IPv4 listener, IPv6 listener, or both on the Internet. The PCP client also knows if it has an IPv4 address or IPv6 address configured on one of its interfaces. It takes the union of this knowledge to decide to which of its PCP servers to send the request (e.g., an IPv4 address or an IPv6 address), and if to send one or two MAP requests for each of its interfaces (e.g., if the PCP client has only an IPv4 address but wants both IPv6 and IPv4 listeners, it sends a MAP request containing the all-zeros IPv6 address in the Suggested External Address field, and sends a second MAP request containing the all-zeros IPv4 address in the Suggested External Address field. If the PCP client has both an IPv4 and IPv6 address, and only wants an IPv4 listener, it sends one MAP request from its IPv4 address (if the PCP server supports NAT44 or IPv4 firewall) or one MAP request from its IPv6 address (if the PCP server supports NAT64)). The PCP client can simply request the desired mapping to determine if the PCP server supports the desired mapping. Applications that embed IP addresses in payloads (e.g., FTP, SIP) will find it beneficial to avoid address family translation, if possible.

The MAP and PEER requests include a Suggested External IP Address field. Some PCP-controlled devices, especially CGN but also multi-homed NPTv6 networks, have a pool of public-facing IP addresses. PCP allows the client to indicate if it wants a mapping assigned on a specific address of that pool or any address of that pool. Some applications will break if mappings are created on different IP addresses, so applications should carefully consider the implications of using this capability. Static mappings for that Internal Address (e.g., those created by a command-line interface on the PCP server or PCP-controlled device) SHOULD also be assigned to the same External Address. Once an Internal Address has no implicit dynamic mappings and no explicit dynamic mappings in the PCP-controlled device, a subsequent implicit or explicit mapping for that Internal Address MAY be assigned to a different External Address. Generally, this re-assignment would occur when a CGN device is load balancing newly-seen Internal Addresses to its public pool of External Addresses.

The following table summarizes how various common PCP deployments use IPv6 and IPv4 addresses. The 'internal' address is implicitly the same as the source IP address of the PCP request, except when the THIRD_PARTY option is used. The 'external' address is the Suggested External Address field of the MAP or PEER request, and its address family is usually the same as the 'internal' address family, except when technologies like NAT64 are used. The 'remote peer' address is the Remote Peer IP Address of the PEER request or the FILTER option of the MAP request, and is always the same address family as the 'internal' address, even when NAT64 is used. In NAT64, the IPv6 PCP client is not necessarily aware of the NAT64 or aware of the actual IPv4 address of the remote peer, so it expresses the IPv6 address from its perspective as shown in the table.

	internal	external	remote peer
	-----	-----	-----
IPv4 firewall	IPv4	IPv4	IPv4
IPv6 firewall	IPv6	IPv6	IPv6
NAT44	IPv4	IPv4	IPv4
NAT64	IPv6	IPv4	IPv6
NPTv6	IPv6	IPv6	IPv6

Figure 5: Address Families with MAP and PEER

9.1. For Operating a Server

A host operating a server (e.g., a web server) listens for traffic on a port, but the server never initiates traffic from that port. For this to work across a NAT or a firewall, the host needs to (a) create a mapping from a public IP address and port to itself as described in [Section 10](#) and (b) publish that public IP address and port via some sort of rendezvous server (e.g., DNS, a SIP message, a proprietary protocol). Publishing the public IP address and port is out of scope of this specification. To accomplish (a), the host follows the procedures described in this section.

As normal, the application needs to begin listening on a port. Then, the application constructs a PCP message with the MAP Opcode, with the external address set to the appropriate all-zeroes address, depending on whether it wants a public IPv4 or IPv6 address.

The following pseudo-code shows how PCP can be reliably used to operate a server:

```
/* start listening on the local server port */
int s = socket(...);
bind(s, ...);
listen(s, ...);

getsockname(s, &internal_sockaddr, ...);
bzero(&external_sockaddr, sizeof(external_sockaddr));

while (1)
{
    /* Note: the "time_to_send_pcp_request()" check below includes:
     * 1. Sending the first request
     * 2. Retransmitting requests due to packet loss
     * 3. Resending a request due to impending lease expiration
     * 4. Resending a request due to server state loss
     * The PCP packet sent is identical in all cases, apart from the
     * Suggested External Address and Port which may differ between
     * (1), (2), and (3).
     */
    if (time_to_send_pcp_request())
        pcp_send_map_request(internal_sockaddr.sin_port,
                             internal_sockaddr.sin_addr,
                             &external_sockaddr, /* will be zero the first time */
                             requested_lifetime, &assigned_lifetime);

    if (pcp_response_received())
        update_rendezvous_server("Client Ident", external_sockaddr);

    if (received_incoming_connection_or_packet())
        process_it(s);

    if (other_work_to_do())
        do_it();

    /* ... */

    block_until_we_need_to_do_something_else();
}
```

Figure 6: Pseudo-code for using PCP to operate a server

9.2. For Operating a Symmetric Client/Server

A host operating a client and server on the same port (e.g., Symmetric RTP [[RFC4961](#)] or SIP Symmetric Response Routing (rport) [[RFC3581](#)]) first establishes a local listener, (usually) sends the local and public IP addresses and ports to a rendezvous service (which is out of scope of this document), and initiates an outbound connection from that same source address and same port. To accomplish this, the application uses the procedure described in this section.

An application that is using the same port for outgoing connections as well as incoming connections MUST first signal its operation of a server using the PCP MAP Opcode, as described in [Section 10](#), and receive a positive PCP response before it sends any packets from that port.

Discussion: In general, a PCP client doesn't know in advance if it is behind a NAT or firewall. On detecting the host has connected to a new network, the PCP client can attempt to request a mapping using PCP, and if that succeeds then the client knows it has successfully created a mapping. If after multiple retries it has received no PCP response, then either the client is **not** behind a NAT or firewall and has unfettered connectivity, or the client **is** behind a NAT or firewall which doesn't support PCP (and the client may still have working connectivity by virtue of static mappings previously created manually by the user). Retransmitting PCP requests multiple times before giving up and assuming unfettered connectivity adds delay in that case. Initiating outbound TCP connections immediately without waiting for PCP avoids this delay, and will work if the NAT has endpoint-independent mapping (EIM) behavior, but may fail if the NAT has endpoint-dependent mapping (EDM) behavior. Waiting enough time to allow an explicit PCP MAP Mapping to be created (if possible) first ensures that the same External Port will then be used for all subsequent TCP SYNs sent from the specified Internal Address and Port. PCP supports both EIM and EDM NATs, so clients need to assume they may be dealing with an EDM NAT. In this case, the client will experience more reliable connectivity if it attempts explicit PCP MAP requests first, before initiating any outbound TCP connections from that Internal Address and Port. See also [Section 14.1](#).

The following pseudo-code shows how PCP can be used to operate a symmetric client and server:

```
/* start listening on the local server port */
int s = socket(...);
bind(s, ...);
listen(s, ...);

getsockname(s, &internal_sockaddr, ...);
bzero(&external_sockaddr, sizeof(external_sockaddr));

while (1)
{
    /* Note: the "time_to_send_pcp_request()" check below includes:
     * 1. Sending the first request
     * 2. Retransmitting requests due to packet loss
     * 3. Resending a request due to impending lease expiration
     * 4. Resending a request due to server state loss
     * The PCP packet sent is identical in all cases, apart from the
     * Suggested External Address and Port which may differ between
     * (1), (2), and (3).
     */
    if (time_to_send_pcp_request())
        pcp_send_map_request(internal_sockaddr.sin_port,
                             internal_sockaddr.sin_addr,
                             &external_sockaddr, /* will be zero the first time */
                             requested_lifetime, &assigned_lifetime);

    if (pcp_response_received())
        update_rendezvous_server("Client Ident", external_sockaddr);

    if (received_incoming_connection_or_packet())
        process_it(s);

    if (need_to_make_outgoing_connection())
        make_outgoing_connection(s, ...);

    if (data_to_send())
        send_it(s);

    if (other_work_to_do())
        do_it();

    /* ... */

    block_until_we_need_to_do_something_else();
}
```


Figure 7: Pseudo-code for using PCP to operate a symmetric client/
server

9.3. For Reducing NAT Keepalive Messages

A host operating a client (e.g., XMPP client, SIP client) sends from a port, and may receive responses, but never accepts incoming connections from other Remote Peers on this port. It wants to ensure the flow to its Remote Peer is not terminated (due to inactivity) by an on-path NAT or firewall. To accomplish this, the application uses the procedure described in this section.

Middleboxes such as NATs or firewalls need to see occasional traffic or will terminate their session state, causing application failures. To avoid this, many applications routinely generate keepalive traffic for the primary (or sole) purpose of maintaining state with such middleboxes. Applications can reduce such application keepalive traffic by using PCP.

Note: For reasons beyond NAT, an application may find it useful to perform application-level keepalives, such as to detect a broken path between the client and server, keep state alive on the Remote Peer, or detect a powered-down client. These keepalives are not related to maintaining middlebox state, and PCP cannot do anything useful to reduce those keepalives.

To use PCP for this function, the application first connects to its server, as normal. Afterwards, it issues a PCP request with the PEER Opcode as described in [Section 11](#).

The following pseudo-code shows how PCP can be reliably used with a dynamic socket, for the purposes of reducing application keepalive messages:

```
int s = socket(...);
connect(s, &remote_peer, ...);

getsockname(s, &internal_sockaddr, ...);
bzero(&external_sockaddr, sizeof(external_sockaddr));

while (1)
{
    /* Note: the "time_to_send_pcp_request()" check below includes:
     * 1. Sending the first request
     * 2. Retransmitting requests due to packet loss
     * 3. Resending a request due to impending lease expiration
     * 4. Resending a request due to server state loss
     * The PCP packet sent is identical in all cases, apart from the
     * Suggested External Address and Port which may differ between
     * (1), (2), and (3).
     */
    if (time_to_send_pcp_request())
        pcp_send_peer_request(internal_sockaddr.sin_port,
                               internal_sockaddr.sin_addr,
                               &external_sockaddr, /* will be zero the first time */
                               remote_peer, requested_lifetime, &assigned_lifetime);

    if (data_to_send())
        send_it(s);

    if (other_work_to_do())
        do_it();

    /* ... */

    block_until_we_need_to_do_something_else();
}
```

Figure 8: Pseudo-code using PCP with a dynamic socket

9.4. For Restoring Lost Implicit TCP Dynamic Mapping State

After a NAT loses state (e.g., because of a crash or power failure), it is useful for clients to re-establish TCP mappings on the NAT. This allows servers on the Internet to see traffic from the same IP address and port, so that sessions can be resumed exactly where they were left off. This can be useful for long-lived connections (e.g., instant messaging) or for connections transferring a lot of data

(e.g., FTP). This can be accomplished by first establishing a TCP connection normally and then sending a PEER request/response and remembering the External Address and External Port. Later, when the NAT has lost state, the client can send a PEER request with the Suggested External Port and Suggested External Address remembered from the previous session, which will create a mapping in the NAT that functions exactly as an implicit dynamic mapping. The client then resumes sending TCP data to the server.

Note: This procedure works well for TCP, provided the NAT creates a new implicit dynamic outbound mapping only for TCP segments with the SYN bit set (i.e., the newly-booted NAT drops the re-transmitted data segments from the client because the NAT does not have an active mapping for those segments), and if the server is not sending data that elicits a RST from the NAT. This is not the case for UDP, because a new UDP mapping will be created (probably on a different port) as soon as UDP traffic is seen by the NAT.

10. MAP Opcode

This section defines an Opcode which controls forwarding from a NAT (or firewall) to an Internal Host.

MAP: Create an explicit dynamic mapping between an Internal Address + Port and an External Address + Port.

PCP Servers SHOULD provide a configuration option to allow administrators to disable MAP support if they wish.

Mappings created by PCP MAP requests are, by definition, Endpoint Independent Mappings (EIM) with Endpoint Independent Filtering (EIF) (unless the FILTER Option is used), even on a NAT that usually creates Endpoint Dependent Mappings (EDM) or Endpoint Dependent Filtering (EDF) for outgoing connections, since the purpose of an (unfiltered) MAP mapping is to receive inbound traffic from any remote endpoint, not from only one specific remote endpoint.

Note also that all NAT mappings (created by PCP or otherwise) are by necessity bidirectional and symmetric. For any packet going in one direction (in or out) that is translated by the NAT, a reply going in the opposite direction needs to have the corresponding opposite translation done so that the reply arrives at the right endpoint. This means that if a client creates a MAP mapping, and then later sends an outgoing packet using the mapping's internal source port, the NAT should translate that packet's Internal Address and Port to the mapping's External Address and Port, so that replies addressed to the External Address and Port are correctly translated to the

mapping's Internal Address and Port.

On Operating Systems that allow multiple listening clients to bind to the same Internal Port, clients MUST ensure that they have exclusive use of that Internal Port (e.g., by binding the port using `INADDR_ANY`, or using `SO_EXCLUSIVEADDRUSE` or similar) before sending their MAP request, to ensure that no other clients on the same machine are also listening on the same Internal Port.

As a side-effect of creating a mapping, ICMP messages associated with the mapping MUST be forwarded (and also translated, if appropriate) for the duration of the mapping's lifetime. This is done to ensure that ICMP messages can still be used by hosts, without application programmers or PCP client implementations needing to use PCP separately to create ICMP mappings for those flows.

The operation of the MAP Opcode is described in this section.

[10.1.](#) MAP Operation Packet Formats

The MAP Opcode has a similar packet layout for both requests and responses. If the Assigned External IP address and Assigned External Port in the PCP response always match the Internal IP Address and Port in the PCP request, then the functionality is purely a firewall; otherwise it pertains to a network address translator which might also perform firewall-like functions.

The following diagram shows the format of the Opcode-specific information in a request for the MAP Opcode.

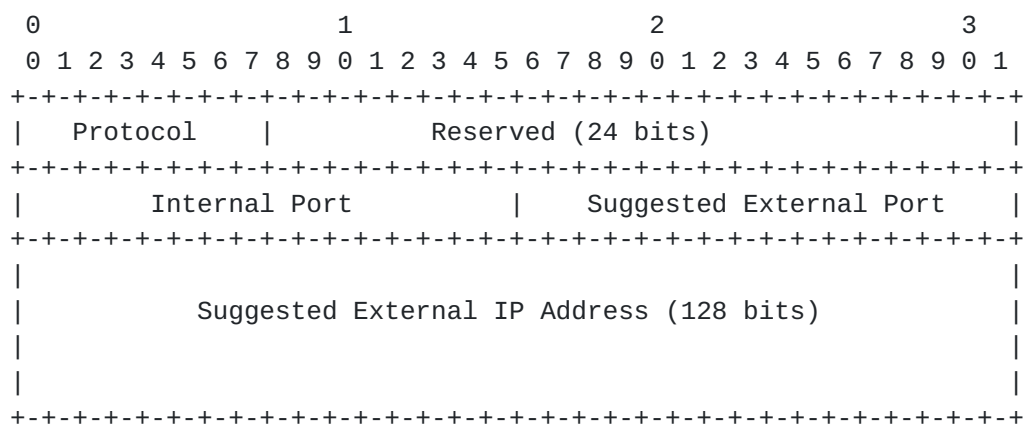


Figure 9: MAP Opcode Request Packet Format

These fields are described below:

Requested lifetime (in common header): Requested lifetime of this mapping, in seconds. The value 0 indicates "delete".

Protocol: Upper-layer protocol associated with this Opcode. Values are taken from the IANA protocol registry [[proto numbers](#)]. For example, this field contains 6 (TCP) if the Opcode is intended to create a TCP mapping. The value 0 has a special meaning for 'all protocols'.

Reserved: 24 reserved bits, MUST be sent as 0 and MUST be ignored when received.

Internal Port: Internal port for the mapping. The 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 Protocol is 0 (meaning 'all protocols')

Suggested External Port: Suggested external port for the mapping. This is useful for refreshing a mapping, especially after the PCP server loses state. If the PCP client does not know the external port, or does not have a preference, it MUST use 0.

Suggested External IP Address: Suggested external IPv4 or IPv6 address. This is useful for refreshing a mapping, especially after the PCP server loses state. If the PCP client does not know the external address, or does not have a preference, it MUST use the address-family-specific all-zeroes address (see [Section 5](#)).

The internal address for the request is the source IP address of the PCP request message itself, unless the THIRD_PARTY Option is used.

The following diagram shows the format of Opcode-specific information in a response packet for the MAP Opcode:

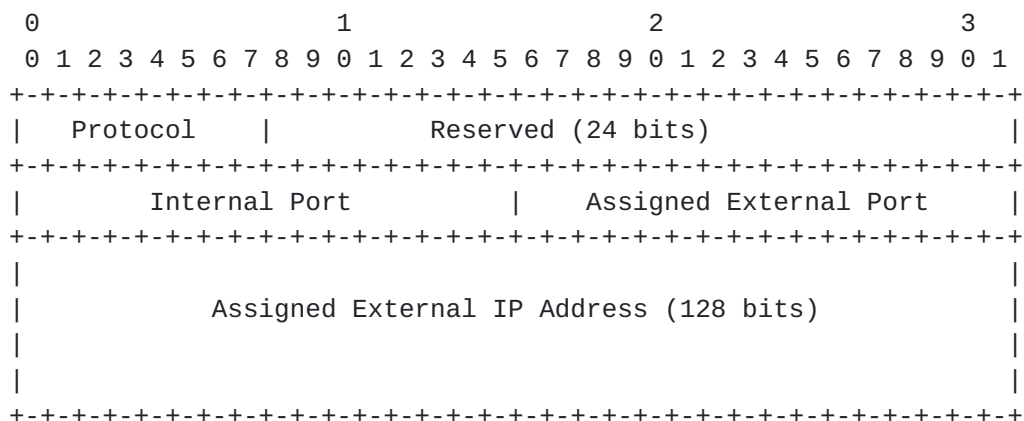


Figure 10: MAP Opcode Response Packet Format

These fields are described below:

Lifetime (in common header): On an error response, this indicates how long clients should assume they'll get the same error response from the PCP server if they repeat the same request. On a success response, this indicates the lifetime for this mapping, in seconds.

Protocol: Copied from the request.

Reserved: 24 reserved bits, MUST be sent as 0 and MUST be ignored when received.

Internal Port: Copied from the request.

Assigned External Port: On a success response, this is the assigned external port for the mapping. On an error response, the Suggested External Port is copied from the request.

Assigned External IP Address: On a success response, this is the assigned external IPv4 or IPv6 address for the mapping. An IPv4 address is encoded using IPv4-mapped IPv6 address. On an error response, the Suggested External IP Address is copied from the request.

10.2. Generating a MAP Request

This section and [Section 10.5](#) describe the operation of a PCP client when sending requests with the MAP Opcode.

The request MAY contain values in the Suggested External Port and Suggested External IP Address fields. This allows the PCP client to attempt to rebuild lost state on the PCP server, which improves the chances of existing connections surviving, and helps the PCP client avoid having to change information maintained at its rendezvous server. Of course, due to other activity on the network (e.g., by other users or network renumbering), the PCP server may not be able to grant the suggested External IP Address and Port, and in that case it will assign a different External IP Address and Port.

If the Protocol does not use 16-bit port numbers (e.g., RSVP), the port number MUST be 0. This will cause all traffic matching that protocol to be mapped.

If the client wants all protocols mapped it uses Protocol 0 (zero) and Internal Port 0 (zero).

10.2.1. Renewing a Mapping

An existing mapping can have its lifetime extended by the PCP client. To do this, the PCP client sends a new MAP request indicating the internal port. The PCP MAP request SHOULD also include the currently assigned external IP address and port in the Suggested External IP address and Suggested External Port fields, so if the PCP server has lost state it can recreate the lost mapping with the same parameters.

The PCP client SHOULD renew the mapping before its expiry time, otherwise it will be removed by the PCP server (see [Section 10.5](#)). To reduce the risk of inadvertent synchronization of renewal requests, a random jitter component should be included. It is RECOMMENDED that PCP clients send a single renewal request packet at a time chosen with uniform random distribution in the range $1/2$ to $5/8$ of expiration time. If no SUCCESS response is received, then the next renewal request should be sent $3/4$ to $3/4 + 1/16$ to expiration, and then another $7/8$ to $7/8 + 1/32$ to expiration, and so on, subject to the constraint that renewal requests MUST NOT be sent less than four seconds apart (a PCP client MUST NOT send a flood of ever-closer-together requests in the last few seconds before a mapping expires).

10.3. Processing a MAP Request

This section and [Section 10.5](#) describe the operation of a PCP server when processing a request with the MAP Opcode. Processing SHOULD be performed in the order of the following paragraphs.

The Protocol and Internal Port fields from the MAP request are copied into the MAP response. If present and processed by the PCP server the THIRD_PARTY Option is also copied into the MAP response.

If the Requested Lifetime is non-zero, it indicates a request to create a mapping or extend the lifetime of an existing mapping. If the PCP server or PCP-controlled device does not support the Protocol, it MUST generate an UNSUPP_PROTOCOL error. If the requested Lifetime is non-zero, the Internal Port is zero, and the Protocol is non-zero, it indicates a request to map all incoming traffic for that entire Protocol. If this request cannot be fulfilled in its entirety, the error UNSUPP_PROTOCOL MUST be returned. If the requested Lifetime is non-zero, the Internal Port is zero, and the Protocol is zero, it indicates a request to map all incoming traffic for all protocols. If this request cannot be fulfilled in its entirety, the error UNSUPP_PROTOCOL MUST be returned. If the Protocol is zero but the Internal Port is non-zero, the error MALFORMED_REQUEST MUST be returned.

If the requested lifetime is zero, it indicates a request to delete an existing mapping or set of mappings. Processing of the lifetime is described in [Section 10.5](#).

If an Option with value less than 128 exists (i.e., mandatory to process) but that Option does not make sense (e.g., the PREFER_FAILURE Option is included in a request with lifetime=0), the request is invalid and generates a MALFORMED_OPTION error.

If the PCP-controlled device is stateless (that is, it does not establish any per-flow state, and simply rewrites the address and/or port in a purely algorithmic fashion), the PCP server simply returns an answer indicating the external IP address and port yielded by this stateless algorithmic translation. This allows the PCP client to learn its external IP address and port as seen by remote peers. Examples of stateless translators include stateless NAT64, 1:1 NAT44, and NPTv6 [[RFC6296](#)], all of which modify addresses but not port numbers.

It is possible that a mapping might already exist for a requested Internal Address and Port. If so, the PCP server takes the following actions:

1. If the MAP request contains the PREFER_FAILURE Option, but the Suggested External Address and Port do not match the External Address and Port of the existing mapping, the PCP server MUST return CANNOT_PROVIDE_EXTERNAL.
2. If the existing mapping is static (created outside of PCP), the PCP server MUST return the External Address and Port of the existing mapping in its response and SHOULD indicate a Lifetime of $2^{32}-1$ seconds, regardless of the Suggested External Address and Port in the request.
3. If the existing mapping is explicit dynamic inbound (created by a previous MAP request), the PCP server MUST return the existing External Address and Port in its response, regardless of the Suggested External Address and Port in the request. Additionally, the PCP server MUST update the lifetime of the existing mapping, in accordance with [section 10.5](#).
4. If the existing mapping is dynamic outbound (created by outgoing traffic or a previous PEER request), the PCP server SHOULD create a new explicit inbound mapping, replicating the ports and addresses from the outbound mapping (but the outbound mapping continues to exist, and remains in effect if the explicit inbound mapping is later deleted).

If no mapping exists for the Internal Address and Port, and the PCP server is able to create a mapping using the Suggested External Address and Port, it SHOULD do so. This is beneficial for re-establishing state lost in the PCP server (e.g., due to a reboot). If the PCP server cannot assign the Suggested External Address and Port but can assign some other External Address and Port (and the request did not contain the PREFER_FAILURE Option) the PCP server MUST do so and return the newly assigned External Address and Port in the response. Cases where a PCP server or PCP-controlled device cannot assign the Suggested External Address and Port include:

- o The Suggested External Address and Port is already assigned to another existing explicit, implicit, or static mapping (i.e., is already forwarding traffic to some other internal address and port).
- o The Suggested External Address and Port is already used by the NAT gateway for one of its own services (e.g., port 80 for the NAT gateway's own configuration pages).
- o The Suggested External Address and Port is otherwise prohibited by the PCP server's policy.
- o The Suggested External Address or port is invalid (e.g., 127.0.0.1, ::1, multicast address, or the port is not valid for the indicated protocol).
- o The Suggested External Address does not belong to the NAT gateway.
- o The Suggested External Address is not configured to be used as an external address of the firewall or NAT gateway.
- o The PREFER_FAILURE option is included in the request and the Suggested External Address and Port are not assignable to the PCP client, which returns the CANNOT_PROVIDE_EXTERNAL error.

By default, a PCP-controlled device MUST NOT create mappings for a protocol not indicated in the request. For example, if the request was for a TCP mapping, a UDP mapping MUST NOT be created.

Mappings typically consume state on the PCP-controlled device, and it is RECOMMENDED that a per-host and/or per-subscriber limit be enforced by the PCP server to prevent exhausting the mapping state. If this limit is exceeded, the result code USER_EX_QUOTA is returned.

If all of the preceding operations were successful (did not generate an error response), then the requested mapping is created or refreshed as described in the request and a SUCCESS response is

built.

[10.4.](#) Processing a MAP Response

This section describes the operation of the PCP client when it receives a PCP response for the MAP Opcode.

After performing common PCP response processing, the response is further matched with a previously-sent MAP request by comparing the Internal IP Address (the destination IP address of the PCP response, or other IP address specified via the THIRD_PARTY option), the Protocol and the Internal Port. Other fields are not compared, because the PCP server sets those fields. Note that if the PCP server supports Mapping Update ([Section 13.2](#)) the PCP server will send additional MAP responses if the mapping changes (e.g., due to IP renumbering).

On a success response, the PCP client can use the External IP Address and Port as needed. Typically the PCP client will communicate the External IP Address and Port to another host on the Internet using an application-specific rendezvous mechanism such as DNS SRV records.

As long as renewal is desired, the PCP client MUST also set a timer or otherwise schedule an event to renew the mapping before its lifetime expires. Renewing a mapping is performed by sending another MAP request, exactly as described in [Section 10.2](#), except that the Suggested External Address and Port SHOULD be set to the values received in the response. From the PCP server's point of view a MAP request to renew a mapping is identical to a MAP request to create a new mapping, and is handled identically. Indeed, in the event of PCP server state loss, a renewal request from a PCP client will appear to the server to be a request to create a new mapping, with a particular Suggested External Address and Port, which happens to be what the PCP server previously assigned. See also [Section 14.3.1](#).

On an error response, the client SHOULD NOT repeat the same request to the same PCP server within the lifetime returned in the response.

[10.5.](#) Mapping Lifetime and Deletion

The PCP client requests a certain lifetime, and the PCP server responds with the assigned lifetime. The PCP server MAY grant a lifetime smaller or larger than the requested lifetime. The PCP server SHOULD be configurable for permitted minimum and maximum lifetime, and the RECOMMENDED values are 120 seconds for the minimum value and 24 hours for the maximum. It is RECOMMENDED that the server be configurable to restrict lifetimes to less than 24 hours, because mappings will consume ports even if the Internal Host is no

longer interested in receiving the traffic or is no longer connected to the network. These recommendations are not strict, and deployments should evaluate the trade offs to determine their own minimum and maximum lifetime values.

Once a PCP server has responded positively to a MAP request for a certain lifetime, the port mapping is active for the duration of the lifetime unless the lifetime is reduced by the PCP client (to a shorter lifetime or to zero) or until the PCP server loses its state (e.g., crashes). Mappings created by PCP MAP requests are not special or different from mappings created in other ways. In particular, it is implementation-dependent if outgoing traffic extends the lifetime of such mappings beyond the PCP-assigned lifetime. PCP clients **MUST NOT** depend on this behavior to keep mappings active, and **MUST** explicitly renew their mappings as required by the Lifetime field in PCP response messages.

Upon receipt of a MAP or PEER response with an absurdly long Assigned Lifetime the PCP client **SHOULD** behave as if it received a more sane value (e.g., 24 hours), and renew the mapping accordingly, to ensure that if the static mapping is removed the client will continue to maintain the mapping it desires.

If the requested lifetime is zero then:

- o If both the internal port and protocol are non-zero, it indicates a request to delete the indicated mapping immediately.
- o If both the internal port and protocol are zero, it indicates a request to delete all mappings for this Internal Address for all transport protocols. This is useful when a host reboots or joins a new network, to clear out prior stale state from the NAT gateway before beginning to install new mappings.
- o If the internal port is zero and the protocol is non-zero, it indicates a request to delete a previous 'wildcard' (all-ports) mapping for that protocol.
- o If the internal port is non-zero and the protocol is zero, then the request is invalid and the PCP Server **MUST** return a **MALFORMED_REQUEST** error to the client.

In requests where the requested Lifetime is 0, the Suggested External Address and Suggested External Port fields **MUST** be set to zero on transmission and **MUST** be ignored on reception, and these fields **MUST** be copied into the Assigned External IP Address and Assigned External Port of the response.

PCP PEER requests cannot delete or shorten lifetimes of any mappings. If a PEER request matches an existing mapping, and the requested lifetime is less than that of the existing mapping, the PCP server returns SUCCESS with the lifetime of the existing mapping.

PCP MAP requests can only delete or shorten lifetimes of MAP-created mappings. If the PCP client attempts to delete a static mapping (i.e., a mapping created outside of PCP itself), or an outbound (implicit or PEER-created) mapping, the PCP server MUST return NOT_AUTHORIZED. If the PCP client attempts to delete a mapping that does not exist, the SUCCESS result code is returned (this is necessary for PCP to be idempotent). If the PCP MAP request was for port=0 (indicating 'all ports'), the PCP server deletes all of the explicit dynamic mappings it can (but not any implicit or static mappings), and returns a SUCCESS response. If the deletion request was properly formatted and successfully processed, a SUCCESS response is generated with lifetime of 0 and the server copies the protocol and internal port number from the request into the response. An inbound mapping (i.e., static mapping or MAP- created dynamic mapping) MUST NOT have its lifetime reduced by transport protocol messages (e.g., TCP RST, TCP FIN). Note the THIRD_PARTY Option, if authorized, can also delete PCP-created mappings (see [Section 12.1](#)).

An application that forgets its PCP-assigned mappings (e.g., the application or OS crashes) will request new PCP mappings. This may consume port mappings, if the application binds to a different Internal Port every time it runs. The application will also likely initiate new implicit dynamic outbound mappings without using PCP, which will also consume port mappings. If there is a port mapping quota for the Internal Host, frequent restarts such as this may exhaust the quota. PCP provides some protections against such port consumption: When a PCP client first acquires a new IP address (e.g., reboots or joins a new network), it SHOULD remove mappings that may already be instantiated for that new Internal Address. To do this, the PCP client sends a MAP request with protocol, internal port, and lifetime set to 0. Some port mapping APIs (e.g., the "DNSServiceNATPortMappingCreate" API provided by Apple's Bonjour on Mac OS X, iOS, Windows, Linux [[Bonjour](#)]) automatically monitor for process exit (including application crashes) and automatically send port mapping deletion requests if the process that requested them goes away without explicitly relinquishing them.

To reduce unwanted traffic and data corruption, External UDP and TCP ports SHOULD NOT be re-used for an interval (TIME_WAIT interval [[RFC0793](#)]). However, the PCP server SHOULD allow the previous user of an External Port to re-acquire the same port during that interval.

10.6. Address Change Events

A customer premises router might obtain a new External IP address, for a variety of reasons including a reboot, power outage, DHCP lease expiry, or other action by the ISP. If this occurs, traffic forwarded to the host's previous address might be delivered to another host which now has that address. This affects both implicit dynamic mappings and explicit dynamic mappings. However, this same problem already occurs today when a host's IP address is re-assigned, without PCP and without an ISP-operated CGN. The solution is the same as today: the problems associated with host renumbering are caused by host renumbering and are eliminated if host renumbering is avoided. PCP defined in this document does not provide machinery to reduce the host renumbering problem.

When an Internal Host changes its IP address (e.g., by having a different address assigned by the DHCP server) the NAT (or firewall) will continue to send traffic to the old IP address. Typically, the Internal Host will no longer receive traffic sent to that old IP address. Assuming the Internal Host wants to continue receiving traffic, it needs to install new mappings for its new IP address. The suggested external port field will not be fulfilled by the PCP server, in all likelihood, because it is still being forwarded to the old IP address. Thus, a mapping is likely to be assigned a new external port number and/or public IP address. Note that such host renumbering is not expected to happen routinely on a regular basis for most hosts, since most hosts renew their DHCP leases before they expire (or re-request the same address after reboot) and most DHCP servers honor such requests and grant the host the same address it was previously using before the reboot.

A host might gain or lose interfaces while existing mappings are active (e.g., Ethernet cable plugged in or removed, joining/leaving a WiFi network). Because of this, if the PCP client is sending a PCP request to maintain state in the PCP server, it SHOULD ensure those PCP requests continue to use the same interface (e.g., when refreshing mappings). If the PCP client is sending a PCP request to create new state in the PCP server, it MAY use a different source interface or different source address.

10.7. Learning the External IP Address Alone

NAT-PMP [[I-D.cheshire-nat-pmp](#)] includes a mechanism to allow clients to learn the External IP Address alone, without also requesting a port mapping. In the case of PCP, this operation no longer makes sense. PCP supports Large Scale NATs (CGN) which may have a pool of External IP Addresses, not just one. A client may not be assigned any particular External IP Address from that pool until it has at

least one implicit, explicit or static port mapping, and even then only for as long as that mapping remains valid. Client software that just wishes to display the user's External IP Address for cosmetic purposes can achieve that by requesting a short-lived mapping (e.g., to the Discard service (TCP/9 or UDP/9) or some other port) and then displaying the resulting External IP Address. However, once that mapping expires a subsequent implicit or explicit dynamic mapping might be mapped to a different external IP address.

11. PEER Opcode

This section defines an Opcode for controlling dynamic mappings.

PEER: Create a new dynamic outbound mapping to a remote peer's IP address and port, or extend the lifetime of an existing outbound mapping.

The use of this Opcodes is described in this section.

PCP Servers SHOULD provide a configuration option to allow administrators to disable PEER support if they wish.

Because a mapping created or managed by PEER behaves almost exactly like an implicit dynamic mapping created as a side-effect of a packet (e.g., TCP SYN) sent by the host, mappings created or managed using PCP PEER requests may be Endpoint Independent Mappings (EIM) or Endpoint Dependent Mappings (EDM), with Endpoint Independent Filtering (EIF) or Endpoint Dependent Filtering (EDF), consistent with the existing behavior of the NAT gateway or firewall in question for implicit outbound mappings it creates automatically as a result of observing outgoing traffic from Internal Hosts.

11.1. PEER Operation Packet Formats

The PEER Opcode allows a PCP client to create a new explicit dynamic outbound mapping (which functions similarly to an outbound mapping created implicitly when a host sends an outbound TCP SYN) or to extend the lifetime of an existing outbound mapping.

The following diagram shows the request packet format for the PEER Opcode. This packet format is aligned with the response packet format:

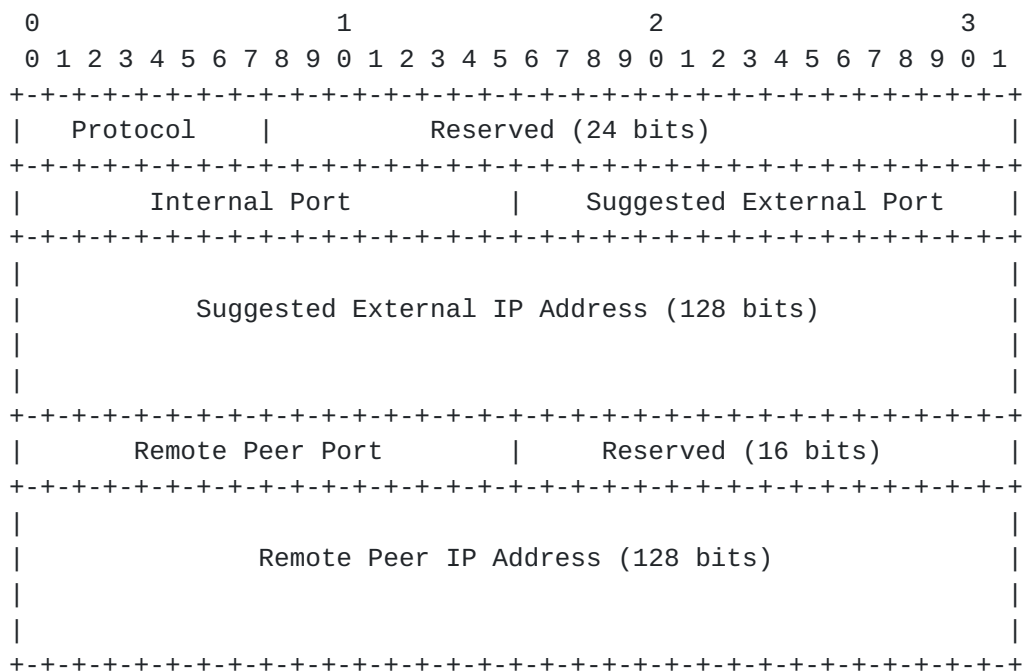


Figure 11: PEER Opcode Request Packet Format

These fields are described below:

Requested Lifetime (in common header): Requested lifetime of this mapping, in seconds. Note that it is not possible to reduce the lifetime of a mapping (or delete it, with requested lifetime=0) using PEER.

Protocol: Upper-layer protocol associated with this Opcode. Values are taken from the IANA protocol registry [[proto numbers](#)]. For example, this field contains 6 (TCP) if the Opcode is describing a TCP mapping.

Reserved: 24 reserved bits, MUST be set to 0 on transmission and MUST be ignored on reception.

Internal Port: Internal port for the mapping.

Suggested External Port: Suggested external port for the mapping. If the PCP client does not know the external port, or does not have a preference, it MUST use 0.

Suggested External IP Address: Suggested External IP Address for the mapping. If the PCP client does not know the external address, or does not have a preference, it MUST use the address-family-specific all-zeroes address (see [Section 5](#)).

Remote Peer Port: Remote peer's port for the mapping.

Reserved: 16 reserved bits, MUST be set to 0 on transmission and MUST be ignored on reception.

Remote Peer IP Address: Remote peer's IP address from the perspective of the PCP client, so that the PCP client does not need to concern itself with NAT64 or NAT46 (which both cause the client's idea of the remote peer's IP address to differ from the remote peer's actual IP address). This field allows the PCP client and PCP server to disambiguate multiple connections from the same port on the Internal Host to different servers. An IPv6 address is represented directly, and an IPv4 address is represented using the IPv4-mapped address syntax ([Section 5](#)).

When attempting to re-create a lost mapping, the Suggested External IP Address and Port are set to the External IP Address and Port fields received in a previous PEER response from the PCP server. On an initial PEER request, the External IP Address and Port are set to zero.

Note that semantics similar to the PREFER_FAILURE option are automatically implied by PEER requests. If the Suggested External IP Address or Suggested External Port fields are non-zero, and the PCP server is unable to honor the Suggested External IP Address or Port, then the PCP server MUST return a CANNOT_PROVIDE_EXTERNAL error response. The PREFER_FAILURE Option is neither required nor allowed in PEER requests, and if PCP server receives a PEER request containing the PREFER_FAILURE Option it MUST return a MALFORMED_REQUEST error response.

The following diagram shows the response packet format for the PEER Opcode:

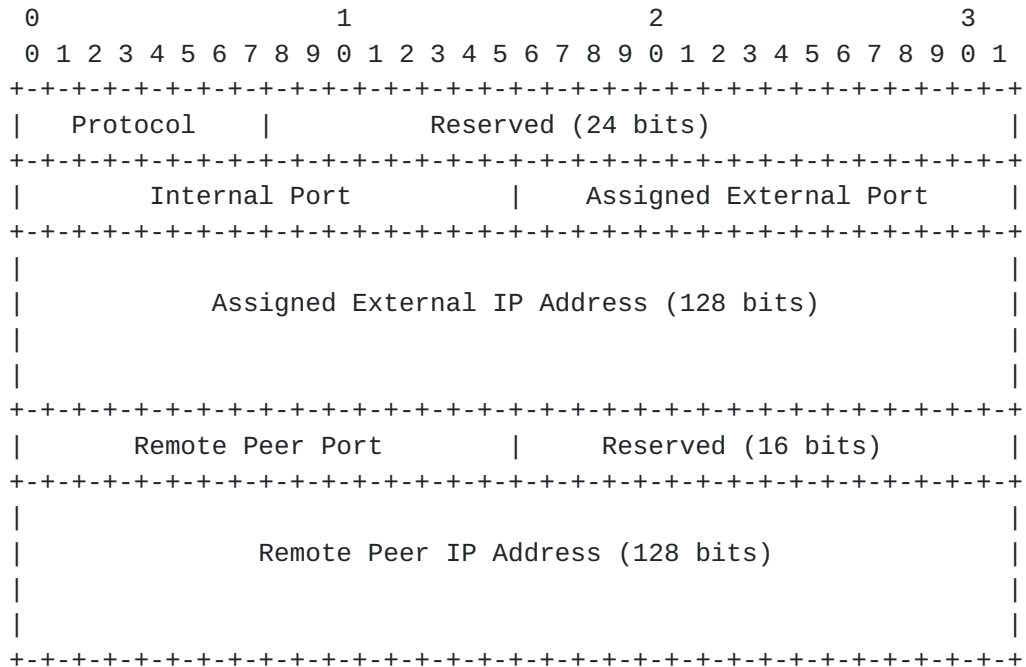


Figure 12: PEER Opcode Response Packet Format

Lifetime (in common header): On a success response, this indicates the lifetime for this mapping, in seconds. On an error response, this indicates how long clients should assume they'll get the same error response from the PCP server if they repeat the same request.

Protocol: Copied from the request.

Reserved: 24 reserved bits, MUST be set to 0 on transmission, MUST be ignored on reception.

Internal Port: Copied from request.

Assigned External Port: On a success response, this is the assigned external port for the mapping. On an error response, the Suggested External Port is copied from the request.

Assigned External IP Address: On a success response, this is the assigned external IPv4 or IPv6 address for the mapping; IPv4 or IPv6 address is indicated by the Opcode. On an error response, the Suggested External IP Address is copied from the request.

Remote Peer port: Copied from request.

Reserved: 16 reserved bits, MUST be set to 0 on transmission, MUST be ignored on reception.

Remote Peer IP Address: Copied from the request.

11.2. Generating a PEER Request

This section describes the operation of a client when generating a message with the PEER Opcode.

The PEER Opcode MAY be sent before or after establishing bi-directional communication with the remote peer.

If sent before, this is considered a PEER-created mapping which creates a new dynamic outbound mapping in the PCP-controlled device. This is useful for restoring a mapping after a NAT has lost its mapping state (e.g., due to a crash).

If sent after, this allows the PCP client to learn the IP address, port, and lifetime of the assigned External Address and Port for the existing implicit dynamic outbound mapping, and potentially to extend this lifetime (for the purpose described in [Section 9.3](#)).

The PEER Opcode contains a Remote Peer Address field, which is always from the perspective of the PCP client. Note that when the PCP-controlled device is performing address family translation (NAT46 or NAT64), the remote peer address from the perspective of the PCP client is different from the remote peer address on the other side of the address family translation device.

11.3. Processing a PEER Request

This section describes the operation of a server when receiving a request with the PEER Opcode. Processing SHOULD be performed in the order of the following paragraphs.

The following fields from a PEER request are copied into the response: Protocol, Internal Port, Remote Peer IP Address, and Remote Peer Port.

When an implicit dynamic mapping is created, some NATs and firewalls validate destination addresses and will not create an implicit dynamic mapping if the destination address is invalid (e.g., 127.0.0.1). If a PCP-controlled device does such validation for implicit dynamic mappings, it SHOULD also do a similar validation of the Remote Peer IP Address and Port for PEER-created explicit dynamic

mappings. If the validation determines the Remote Peer IP Address of a PEER request is invalid, then no mapping is created, and a MALFORMED_REQUEST error result is returned.

On receiving the PEER Opcode, the PCP server examines the mapping table. If the requested mapping does not yet exist, and the Suggested External Address and Port can be honored, the mapping is created. By having PEER create such a mapping, we avoid a race condition between the PEER request or the initial outgoing packet arriving at the NAT or firewall device first, and allow PEER to be used to recreate an outbound dynamic mapping (see last paragraph of [Section 14.3.1](#)). Thereafter, this PEER-created mapping is treated as if it was an implicit dynamic outbound mapping (e.g., as if the PCP client sent a TCP SYN) and a Lifetime appropriate to such a mapping is returned (note: on many NATs and firewalls, such mapping lifetimes are very short until the bi-directional traffic is seen by the NAT or firewall). If the requested mapping does not yet exist, and Suggested External Address and Port cannot be honored, the error CANNOT_PROVIDE_EXTERNAL is returned. If the requested mapping already exists, it is a request to modify the lifetime of that existing mapping.

The PEER Opcode can extend the lifetime of an existing dynamic outbound mapping. The PCP server may grant the client's requested lifetime, or may grant a value higher or lower, depending on local policy. The PEER Opcode MUST NOT ever reduce the lifetime of an existing outbound mapping. If the Requested Lifetime is less than the lifetime of the existing mapping, it is treated as a request for the lifetime of the mapping (this can be used to query the lifetime of an existing mapping).

If all of the preceding operations were successful (did not generate an error response), then a SUCCESS response is generated, with the Lifetime field containing the lifetime of the mapping.

If a PEER-created or PEER-managed mapping is not renewed using PEER, then it reverts to the NAT's usual behavior for implicit mappings, e.g., continued outbound traffic keeps the mapping alive, as per the NAT or firewall device's existing policy. A PEER-created or PEER-managed mapping may be terminated at any time by action of the TCP client or server (e.g., due to TCP FIN or TCP RST), as per the NAT or firewall device's existing policy.

11.4. Processing a PEER Response

This section describes the operation of a client when processing a response with the PEER Opcode.

After performing common PCP response processing, the response is further matched with an outstanding PEER request by comparing the Internal IP Address (the destination IP address of the PCP response, or other IP address specified via the THIRD_PARTY option), the Protocol, the Internal Port, the Remote Peer Address and the Remote Peer Port. Other fields are not compared, because the PCP server sets those fields to provide information about the mapping created by the Opcode. Note that if the PCP server supports Mapping Update ([Section 13.2](#)) the PCP server will send additional PEER responses if the mapping changes (e.g., due to IP renumbering).

On a successful response, the application can use the assigned lifetime value to reduce its frequency of application keepalives for that particular NAT mapping. Of course, there may be other reasons, specific to the application, to use more frequent application keepalives. For example, the PCP assigned lifetime could be one hour but the application may want to maintain state on its server (e.g., "busy" / "away") more frequently than once an hour. If the response indicates an unexpected IP address or port (e.g., due to IP renumbering), the PCP client will want to re-establish its connection to its remote server.

If the PCP client wishes to keep this mapping alive beyond the indicated lifetime, it MAY issue a new PCP request prior to the expiration, or it MAY rely on continued inside-to-outside traffic to ensure the mapping will continue to exist. See [Section 10.2.1](#) for recommended renewal timing.

Note: implementations need to expect the PEER response may contain an External IP Address with a different family than the Remote Peer IP Address, e.g., when NAT64 or NAT46 are being used.

[12.](#) Options for MAP and PEER Opcodes

This section describes Options for the MAP and PEER Opcodes. These Options MUST NOT appear with other Opcodes, unless permitted by those other Opcodes.

[12.1.](#) THIRD_PARTY Option for MAP and PEER Opcodes

This Option is used when a PCP client wants to control a mapping to an Internal Host other than itself. This is used with both MAP and PEER Opcodes.

The THIRD_PARTY Option is formatted as follows:

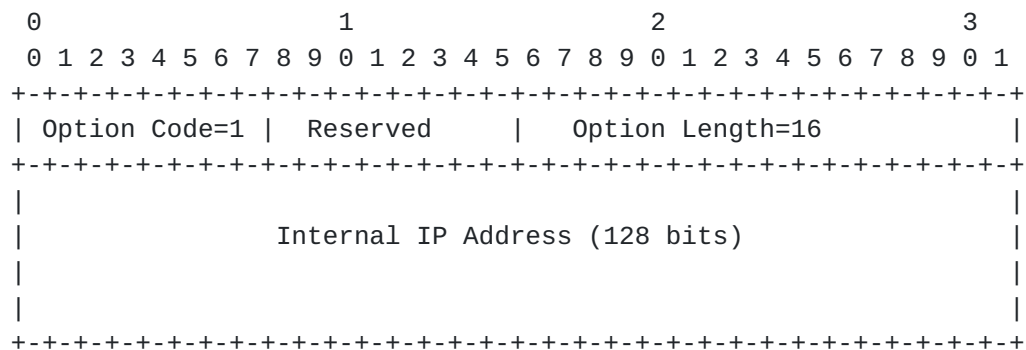


Figure 13: THIRD_PARTY Option packet format

The fields are described below:

Internal IP Address: Internal IP address for this mapping.

Option Name: THIRD_PARTY

Number: 1

Purpose: Indicates the MAP or PEER request is for a host other than the host sending the PCP Option.

Valid for Opcodes: MAP, PEER

Length: 16 octets

May appear in: request. May appear in response only if it appeared in the associated request.

Maximum occurrences: 1

A THIRD_PARTY Option MUST NOT contain the same address as the source address of the packet. A PCP server receiving a THIRD_PARTY Option specifying the same address as the source address of the packet MUST return a MALFORMED_REQUEST result code. This is because many PCP servers may not implement the THIRD_PARTY Option at all, and a client using the THIRD_PARTY Option to specify the same address as the source address of the packet will cause mapping requests to fail where they would otherwise have succeeded.

A PCP server MAY be configured to permit or to prohibit the use of the THIRD_PARTY Option. If this Option is permitted, properly authorized clients may perform these operations on behalf of other hosts. If this Option is prohibited, and a PCP server receives a PCP MAP request with a THIRD_PARTY Option, it MUST generate a UNSUPP_OPTION response.

It is RECOMMENDED that customer premises equipment implementing a PCP Server be configured to prohibit third party mappings by default. With this default, if a user wants to create a third party mapping,

the user needs to interact out-of-band with their customer premises router (e.g., using its HTTP administrative interface).

It is RECOMMENDED that service provider NAT and firewall devices implementing a PCP Server be configured to permit the THIRD_PARTY Option, when sent by a properly authorized host. If the packet arrives from an unauthorized host, the PCP server MUST generate an UNSUPP_OPTION error.

Determining which PCP clients are authorized to use the THIRD_PARTY Option for which other hosts is deployment-dependent. For example, an ISP using Dual-Stack Lite could choose to allow a client connecting over a given IPv6 tunnel to manage mappings for any other host connecting over the same IPv6 tunnel, or the ISP could choose to allow only the DS-Lite B4 element to manage mappings for other hosts connecting over the same IPv6 tunnel. A cryptographic authentication and authorization model is outside the scope of this specification. Note that the THIRD_PARTY Option is not needed for today's common scenario of an ISP offering a single IP address to a customer who is using NAT to share that address locally, since in this scenario all the customer's hosts appear to be a single host from the point of view of the ISP.

Where possible, it may be beneficial if a client using the THIRD_PARTY Option to create and maintain mappings on behalf of some other device can take steps to verify that the other device is still present and active on the network. Otherwise the client using the THIRD_PARTY Option to maintain mappings on behalf of some other device risks maintaining those mappings forever, long after the device that required them has gone. This would defeat the purpose of PCP mappings having a finite lifetime so that they can be automatically deleted after they are no longer needed.

A PCP client can delete all PCP-created explicit dynamic inbound mappings (i.e., those created by PCP MAP requests) that it is authorized to delete by sending a PCP MAP request with zero in the Internal Port field, zero in the Protocol field, and a THIRD_PARTY Option with the all-zeros address in the Internal IP Address field. Upon receipt of such a request from an authorized PCP client, the PCP server MUST delete all described mappings the PCP client is authorized to delete, and return SUCCESS. (For idempotency, SUCCESS is returned if zero or more mappings were deleted.)

12.2. PREFER_FAILURE Option for MAP Opcode

This Option is only used with the MAP Opcode.

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.

The PREFER_FAILURE Option is formatted as follows:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Option Code=2 | Reserved      | Option Length=0                      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Figure 14: PREFER_FAILURE Option packet format

Option Name: PREFER_FAILURE

Number: 2

Purpose: indicates that the PCP server should not create an alternative mapping if the suggested external port and address cannot be mapped.

Valid for Opcodes: MAP

Length: 0

May appear in: request. May appear in response only if it appeared in the associated request.

Maximum occurrences: 1

The result code CANNOT_PROVIDE_EXTERNAL is returned if the Suggested External Address and Port cannot be mapped. This can occur because the External Port is already mapped to another host's outbound dynamic mapping, an inbound dynamic mapping, a static mapping, or the same Internal Address and Port already has an outbound dynamic mapping which is mapped to a different External Port than suggested. This can also occur because the External Address is no longer available (e.g., due to renumbering). The server MAY set the Lifetime in the response to the remaining lifetime of the conflicting mapping, rounded up to the next larger integer number of seconds.

This Option exists for interworking with non-PCP mapping protocols that have different semantics than PCP (e.g., UPnP IGDv1 interworking [[I-D.bpw-pcp-upnp-igd-interworking](#)], where the semantics of UPnP IGDv1 only allow the UPnP IGDv1 client to dictate mapping a specific port), or separate port allocation systems which allocate ports to a subscriber (e.g., a subscriber-accessed web portal operated by the same ISP that operates the PCP server). A PCP server MAY support this Option, if its designers wish to support such downstream devices or separate port allocation systems. PCP servers that are not intended to interface with such systems are not required to support this Option. PCP clients other than UPnP IGDv1 interworking clients or other than a separate port allocation system SHOULD NOT use this

Option because it results in inefficient operation, and they cannot safely assume that all PCP servers will implement it. It is anticipated that this Option will be deprecated in the future as more clients adopt PCP natively and the need for this Option declines.

PCP servers MAY choose to rate-limit their handling of PREFER_FAILURE requests, to protect themselves from a rapid flurry of 65535 consecutive PREFER_FAILURE requests from clients probing to discover which external ports are available.

There can exist a race condition between the MAP Opcode using the PREFER_FAILURE option and Mapping Update ([Section 13.2](#)). Because of this, the PCP client MUST validate that the External IP Address and Port in a success response matches the associated suggested values from the request. If they don't match, it is because the Mapping Update was sent before the MAP request was processed. If the PCP server has no use for this new mapping, it SHOULD delete the mapping.

[12.3](#). FILTER Option for MAP Opcode

This Option is only used with the MAP Opcode.

This Option indicates that filtering incoming packets is desired. The protocol being filtered is indicated by the MAP Opcode, and the Remote Peer Port and Remote Peer IP Address of the FILTER Option indicate the permitted remote peer's source IP address and port for packets from the Internet. The remote peer prefix length indicates the length of the remote peer's IP address that is significant; this allows a single Option to permit an entire subnet. After processing this MAP request containing the FILTER Option and generating a successful response, the PCP-controlled device will drop packets received on its public-facing interface that don't match the filter fields. After dropping the packet, if its security policy allows, the PCP-controlled device MAY also generate an ICMP error in response to the dropped packet.

The FILTER Option is formatted as follows:

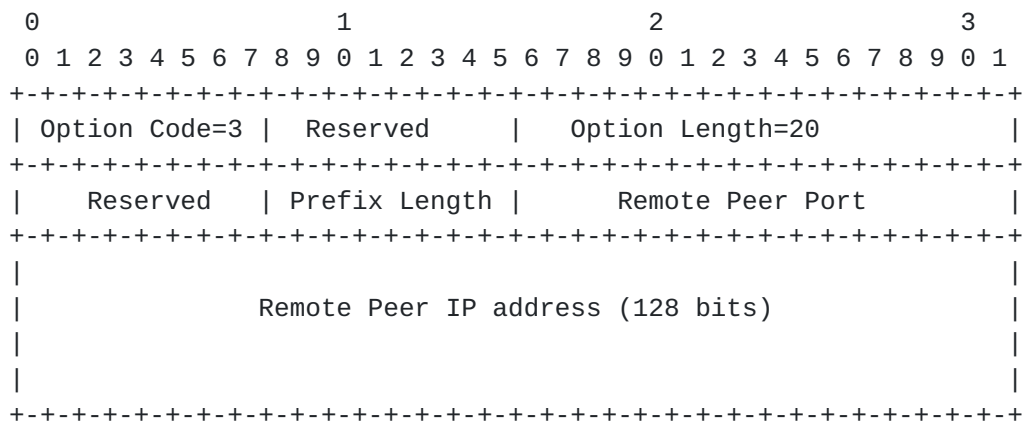


Figure 15: FILTER Option layout

These fields are described below:

Reserved: 8 reserved bits, MUST be sent as 0 and MUST be ignored when received.

Prefix Length: indicates how many bits of the IPv4 or IPv6 address are relevant for this filter. The value 0 indicates "no filter", and will remove all previous filters. See below for detail.

Remote Peer Port: the port number of the remote peer. The value 0 indicates "all ports".

Remote Peer IP address: The IP address of the remote peer.

Option Name: FILTER

Number: 3

Purpose: specifies a filter for incoming packets

Valid for Opcodes: MAP

Length: 20 octets

May appear in: request. May appear in response only if it appeared in the associated request.

Maximum occurrences: as many as fit within maximum PCP message size

The Prefix Length indicates how many bits of the address are used for the filter. For IPv4 addresses (which are encoded using the IPv4-mapped address format (::FFFF:0:0/96)), this means valid prefix lengths are between 96 and 128 bits, inclusive. That is, add 96 to the IPv4 prefix length. For IPv6 addresses, valid prefix lengths are between 0 and 128 bits, inclusive. Values outside those ranges cause the PCP server to return the MALFORMED_OPTION result code.

If multiple occurrences of the FILTER Option exist in the same MAP request, they are processed in the order received (as per normal PCP Option processing) and they MAY overlap the filtering requested. If an existing mapping exists (with or without a filter) and the server receives a MAP request with FILTER, the filters indicated in the new request are added to any existing filters. If a MAP request has a lifetime of 0 and contains the FILTER Option, the error MALFORMED_OPTION is returned.

If any occurrences of the FILTER Option in a request packet are not successfully processed then an error is returned (e.g., MALFORMED_OPTION if one of the Options was malformed) and as with other PCP errors, returning an error causes no state to be changed in the PCP server or in the PCP-controlled device.

To remove all existing filters, the Prefix Length 0 is used. There is no mechanism to remove a specific filter.

To change an existing filter, the PCP client sends a MAP request containing two FILTER Options, the first Option containing a Prefix Length of 0 (to delete all existing filters) and the second containing the new remote peer's IP address and port. Other FILTER Options in that PCP request, if any, add more allowed Remote Peers.

The PCP server or the PCP-controlled device is expected to have a limit on the number of remote peers it can support. This limit might be as small as one. If a MAP request would exceed this limit, the entire MAP request is rejected with the result code EXCESSIVE_REMOTE_PEERS, and the state on the PCP server is unchanged.

All PCP servers MUST support at least one filter per MAP mapping.

The use of the FILTER Option can be seen as a performance optimization. Since all software using PCP to receive incoming connections also has to deal with the case where it may be directly connected to the Internet and receive unrestricted incoming TCP connections and UDP packets, if it wishes to restrict incoming traffic to a specific source address or group of source addresses such software already needs to check the source address of incoming traffic and reject unwanted traffic. However, the FILTER Option is a particularly useful performance optimization for battery powered wireless devices, because it can enable them to conserve battery power by not having to wake up just to reject a unwanted traffic.

13. Rapid Recovery

PCP includes a rapid recovery feature, which allows PCP clients to

repair failed mappings within seconds, rather than the minutes or hours it might take if they relied solely on waiting for the next routine renewal of the mapping. Mapping failures may occur when a NAT gateway is rebooted and loses its mapping state, or when a NAT gateway has its external IP address changed so that its current mapping state becomes invalid.

The PCP rapid recovery feature enables users to, for example, connect to remote machines using ssh, and then reboot their NAT or firewall device (or even replace it with completely new hardware) without losing their established ssh connections.

Use of PCP rapid recovery is a performance optimization to PCP's routine self-healing. Without rapid recovery, PCP clients will still recreate their correct state when they next renew their mappings, but this routine self-healing process may take hours rather than seconds, and will probably not happen fast enough to prevent active TCP connections from timing out.

There are two mechanisms to perform rapid recovery, described below.

13.1. ANNOUNCE Opcode

The rapid recovery mechanism uses the ANNOUNCE Opcode. When the PCP server loses its state (e.g., it lost its state when rebooted), it sends the ANNOUNCE response to a multicast address (specific address explained below) if a multicast network exists on its local interface or, if configured with the IP address of PCP client(s), sends unicast ANNOUNCE responses to those address(es). Additionally, an ANNOUNCE request can be sent (unicast) by a PCP client which elicits a unicast ANNOUNCE response like any other Opcode.

13.1.1. ANNOUNCE operation packet format

The PCP ANNOUNCE Opcode requests and responses have no Opcode-specific payload (that is, the length of the Opcode-specific data is zero), so a packet layout is not shown. The Requested Lifetime field of requests and Lifetime field of responses are both set to 0 on transmission and ignored on reception.

If a PCP server receives an ANNOUNCE request, it first parses it and generates a SUCCESS if parsing and processing of ANNOUNCE is successful (i.e., an error is generated if the request packet is malformed, such as invalid length). Note that, in the future, Options MAY be sent with the PCP ANNOUNCE Opcode; PCP clients and servers need to be prepared to receive Options with the ANNOUNCE Opcode.

Discussion: A separate port is used for server-to-client Restart Announcements (5350) than client-to-server requests (5351) because a single device may have multiple roles. For example, multi-function home gateway may also provide printer sharing, or a home computer may also provide "Internet Sharing" (NAT) functionality. Such devices need to act as both a PCP Server and a PCP Client at the same time, and the software that implements the PCP Server on the device may not be the same software component that implements the PCP Client. The software that implements the PCP Server needs to listen for unicast client requests, whereas the software that implements the PCP Client needs to listen for multicast restart announcements. In many networking APIs it is difficult or impossible to have two independent clients listening for both unicasts and multicasts on the same port at the same time. For this reason, two ports are used.

13.1.2. Generating and Processing a Solicited ANNOUNCE Message

The PCP ANNOUNCE Opcode MAY be sent (unicast) by a PCP client. The Requested Lifetime value MUST be set to zero.

When the PCP server receives the ANNOUNCE Opcode and successfully parses and processes it, it generates SUCCESS response with an Assigned Lifetime of zero.

This functionality allows a PCP client to determine a server's Epoch, or to determine if a PCP server is running, without changing the server's state.

13.1.3. Generating and Processing an Unsolicited ANNOUNCE Message

Generating a PCP restart announcement, described in this section, is an optional feature of PCP. This message is most typically multicast, but can also be unicast. Unicast requires the PCP server send the message to the PCP client's IP address(es), which it may not know after it loses state. When sending unsolicited unicast ANNOUNCE responses, they are sent to port 5351. When sending unsolicited multicast ANNOUNCE responses, they are sent to port 5350. When sending unsolicited responses, the ANNOUNCE Opcode MUST have Result Code equal to zero (SUCCESS).

When a PCP server device that implements Restart Announcement reboots, restarts its NAT engine, or otherwise enters a state where it may have lost some or all of its previous mapping state (or enters a state where it doesn't even know whether it may have had prior state that it lost) it MUST inform PCP clients of this fact by unicasting or multicasting a gratuitous PCP ANNOUNCE Opcode response packet, as shown below, via paths over which it accepts PCP requests.

If sending a multicast ANNOUNCE message, a PCP server device which accepts PCP requests over IPv4 sends the Restart Announcement to the IPv4 multicast address 224.0.0.1:5350 (224.0.0.1 is the All Hosts multicast group address). If sending a multicast ANNOUNCE message, a PCP server device which accepts PCP requests over IPv6 sends the Restart Announcement to the IPv6 multicast address [ff02::1]:5350 (ff02::1 is for all nodes on the local segment). A PCP server device which accepts PCP requests over both IPv4 and IPv6 sends a pair of Restart Announcements, one to each multicast address. If sending unicast ANNOUNCE messages, it sends ANNOUNCE response message to the IP address(es) of the PCP clients with destination port 5350. To accommodate packet loss, the PCP server device MAY transmit such packets (or packet pairs) up to ten times (with an appropriate Epoch time value in each to reflect the passage of time between transmissions) provided that the interval between the first two notifications is at least 250ms, and the interval between subsequent notification at least doubles.

A PCP client that sends PCP requests to a PCP Server via a multicast-capable path, and implements the Restart Announcement feature, and wishes to receive these announcements, MUST listen to receive these PCP Restart Announcements (gratuitous PCP ANNOUNCE Opcode response packets) on the appropriate multicast-capable interfaces on which it sends PCP requests, and MAY listen to receive those on their unicast address. A PCP client device which sends PCP requests using IPv4 listens for packets sent to the IPv4 multicast address 224.0.0.1:5350. A PCP client device which sends PCP requests using IPv6 listens for packets sent to the IPv6 multicast address [ff02::1]:5350. A PCP client device which sends PCP requests using both IPv4 and IPv6 listens for both types of Restart Announcement. (The SO_REUSEPORT socket option or equivalent should be used for the multicast UDP port, if required by the host OS to permit multiple independent listeners on the same multicast UDP port.)

Upon receiving a unicasted or multicasted PCP ANNOUNCE Opcode response packet, a PCP client MUST (as it does with all received PCP response packets) inspect the Announcement's source IP address, and if the Epoch time value is outside the expected range for that server, then for all PCP mappings it made at that server address the client should issue new PCP requests to recreate any lost mapping state. The use of the Suggested External IP Address and Suggested External Port fields in the client's subsequent renewal requests allows the client to remind the restarted PCP server device of what mappings the client had previously been given, so that in many cases the prior state can be recreated. For PCP server devices that reboot relatively quickly it is usually possible to reconstruct lost mapping state fast enough that existing TCP connections and UDP communications do not time out and continue without failure.

13.2. PCP Mapping Update

This rapid recovery mechanism is used when the PCP server remembers its state and determines its existing mappings are invalid (e.g., IP renumbering changes the public IP address of a PCP-controlled NAT).

Implementation of the feature described in this section is optional for PCP servers. It is anticipated that servers which are routinely reconfigured by an administrator or have their WAN address changed frequently will implement this feature (e.g., residential CPE routers). It is anticipated that servers which are not routinely reconfigured will not implement this feature (e.g., service provider-operated CGN).

If a PCP server device has not forgotten its mapping state, but for some other reason has determined that some or all of its mappings have become unusable (e.g., when a home gateway is assigned a different external IPv4 address by the upstream DHCP server) then the PCP server device MAY chose to remedy this situation by automatically repairing its mappings and notifying its clients by following the procedure described below.

For MAP-created and PEER-managed mappings, for each one the PCP server device should update the External IP Address and External Port to appropriate available values, and then send unicast PCP MAP or PEER responses (as appropriate for the mapping) to inform the PCP client of the new External IP Address and External Port. Such MAP responses are identical to the MAP or PEER responses normally returned in response to client MAP or PEER requests containing newly updated External IP Address and External Port values.

Discussion: It could have been possible to design this so that the PCP server (1) sent an ANNOUNCE Opcode to the PCP client, the PCP client reacted by (2) sending a new MAP request and (3) receiving a MAP response. Instead, that design is short-cutted by the server simply sending the message it would have sent in (3).

To accommodate packet loss, the PCP server device MAY transmit such packets up to ten times (with an appropriate Epoch time value in each to reflect the passage of time between transmissions) provided that the interval between the first two notifications is at least 250ms, and the interval between subsequent notification at least doubles.

Upon receipt of such MAP or PEER response, a PCP client uses the information in the response to adjust rendezvous servers or re-connect to servers, respectively. For MAP, this would means updating the DNS entries or other address and port information recorded with some kind of application-specific rendezvous server. For PEER

responses indicating the external port or address changed, this would typically mean re-establishing connections to servers. Any time the external address or port changes, existing TCP and UDP connections will be lost; PCP can't avoid that, but does provide immediate notification of the event to lessen the impact.

14. Implementation Considerations

14.1. Implementing MAP with EDM port-mapping NAT

This section provides non-normative guidance that may be useful to implementers.

For implicit dynamic outbound mappings, some existing NAT devices have endpoint-independent mapping (EIM) behavior while other NAT devices have endpoint-dependent mapping (EDM) behavior. NATs which have EIM behavior do not suffer from the problem described in this section. The IETF strongly encourages EIM behavior [[RFC4787](#)][RFC5382].

In such EDM NAT devices, the same external port may be used by an outbound dynamic mapping and an inbound dynamic mapping (from the same Internal Host or from a different Internal Host). This complicates the interaction with the MAP Opcode. With such NAT devices, there are two ways envisioned to implement the MAP Opcode:

1. Have outbound mappings use a different set of public ports than inbound mappings (e.g., those created with MAP), thus reducing the interaction problem between them; or
2. On arrival of a packet (inbound from the Internet or outbound from an Internal Host), first attempt to use a dynamic outbound mapping to process that packet. If none match, attempt to use an inbound mapping to process that packet. This effectively 'prioritizes' outbound mappings above inbound mappings.

14.2. Lifetime of Explicit and Implicit Dynamic Mappings

This section provides non-normative guidance that may be useful to implementers.

No matter if a NAT is EIM or EDM, it is possible that one (or more) outbound mappings, using the same internal port on the Internal Host, might be created before or after a MAP request. When this occurs, it is important that the NAT honor the Lifetime returned in the MAP response. Specifically, if a mapping was created with the MAP Opcode, the implementation needs to ensure that termination of an

outbound mapping (e.g., via a TCP FIN handshake) does not prematurely destroy the MAP-created inbound mapping.

14.3. PCP Failure Scenarios

This section provides non-normative guidance that may be useful to implementers.

If an event occurs that causes the PCP server to lose dynamic mapping state (such as a crash or power outage), the mappings created by PCP are lost. Occasional loss of state may be unavoidable in a residential NAT device which does not write transient information to non-volatile memory. Loss of state is expected to be rare in a service provider environment (due to redundant power, disk drives for storage, etc.). Of course, due to outright failure of service provider equipment (e.g., software malfunction), state may still be lost.

The Epoch Time allows a client to deduce when a PCP server may have lost its state. When the Epoch Time value is observed to be outside the expected range, the PCP client can attempt to recreate the mappings following the procedures described in this section.

Further analysis of PCP failure scenarios is in [\[I-D.boucadair-pcp-failure\]](#).

14.3.1. Recreating Mappings

This section provides non-normative guidance that may be useful to implementers.

A mapping renewal packet is formatted identically to an original mapping request; from the point of view of the client it is a renewal of an existing mapping, but from the point of view of a newly rebooted PCP server it appears as a new mapping request. In the normal process of routinely renewing its mappings before they expire, a PCP client will automatically recreate all its lost mappings.

When the PCP server loses state and begins processing new PCP messages, its Epoch time is reset and begins counting again. As the result of receiving a packet where the Epoch time field is outside the expected range ([Section 7.5](#)), indicating that a reboot or similar loss of state has occurred, the client can renew its port mappings sooner, without waiting for the normal routine renewal time.

14.3.2. Maintaining Mappings

This section provides non-normative guidance that may be useful to implementers.

A PCP client refreshes a mapping by sending a new PCP request containing information from the earlier PCP response. The PCP server will respond indicating the new lifetime. It is possible, due to reconfiguration or failure of the PCP server, that the public IP address and/or public port, or the PCP server itself, has changed (due to a new route to a different PCP server). Such events are not an error. The PCP server will simply return a new External Address and/or External Port to the client, and the client should record this new External Address and Port with its rendezvous service. To detect such events more quickly, the PCP client may find it beneficial to use shorter lifetimes (so that it communicates with the PCP server more often).

If the PCP client has several mappings, the Epoch Time value only needs to be retrieved for one of them to determine whether or not it appears the PCP server may have suffered a catastrophic loss of state. If the client wishes to check the PCP server's Epoch Time, it sends a PCP request for any one of the client's mappings. This will return the current Epoch Time value. In that request the PCP client could extend the mapping lifetime (by asking for more time) or maintain the current lifetime (by asking for the same number of seconds that it knows are remaining of the lifetime).

If a PCP client changes its Internal IP Address (e.g., because the Internal Host has moved to a new network), and the PCP client wishes to still receive incoming traffic, it needs create new mappings on that new network. New mappings will typically also require an update to the application-specific rendezvous server if the External Address or Port are different to the previous values (see [Section 9.1](#) and [Section 10.6](#)).

14.3.3. SCTP

Although SCTP has port numbers like TCP and UDP, SCTP works differently when behind an address-sharing NAT, in that SCTP port numbers are not changed [[I-D.ietf-behave-sctpnat](#)]. Outbound dynamic SCTP mappings use the verification tag of the association instead of the local and remote peer port numbers. As with TCP, explicit outbound mappings can be made to reduce keepalive intervals, and explicit inbound mappings can be made by passive listeners expecting to receive new associations at the external port.

Because an SCTP-aware NAT does not rewrite SCTP port numbers, a PCP

MAP or PEER request for an SCTP mapping SHOULD provide the same Internal Port and Suggested External Port. If the PCP server supports SCTP, and the suggested external port cannot be provided in an explicit dynamic SCTP mapping, then the error CANNOT_PROVIDE_EXTERNAL is returned. This places an extra burden on the SCTP client because it then has to tear down its listening socket and try again with a different Internal Port, repeatedly until it is successful in finding a port it can use.

The SCTP complications described above occur because of address sharing. The SCTP complications are avoided when address sharing is avoided (e.g., 1:1 NAT, firewall).

14.4. Source Address Replicated in PCP Header

All PCP requests include the PCP client's IP address replicated in the PCP header. This is used to detect address rewriting (NAT) between the PCP client and its PCP server. On operating systems that support the sockets API, the following steps are RECOMMENDED for a PCP client to insert the correct source address and port to include in the PCP header:

1. Create a UDP socket.
2. Call "connect" on this UDP socket using the address and port of the desired PCP server.
3. Call the getsockname() function to retrieve a sockaddr containing the source address the kernel will use for UDP packets sent through this socket.
4. If the IP address is an IPv4 address, encode the address into an IPv4-mapped IPv6 address. Place the native IPv6 address or IPv4-mapped IPv6 address into the PCP Client's IP Address field in the PCP header.
5. Send PCP requests using this connected UDP socket.

14.5. State Diagram

Each mapping entry of the PCP-controlled device would go through the state machine shown below. This state diagram is non-normative.

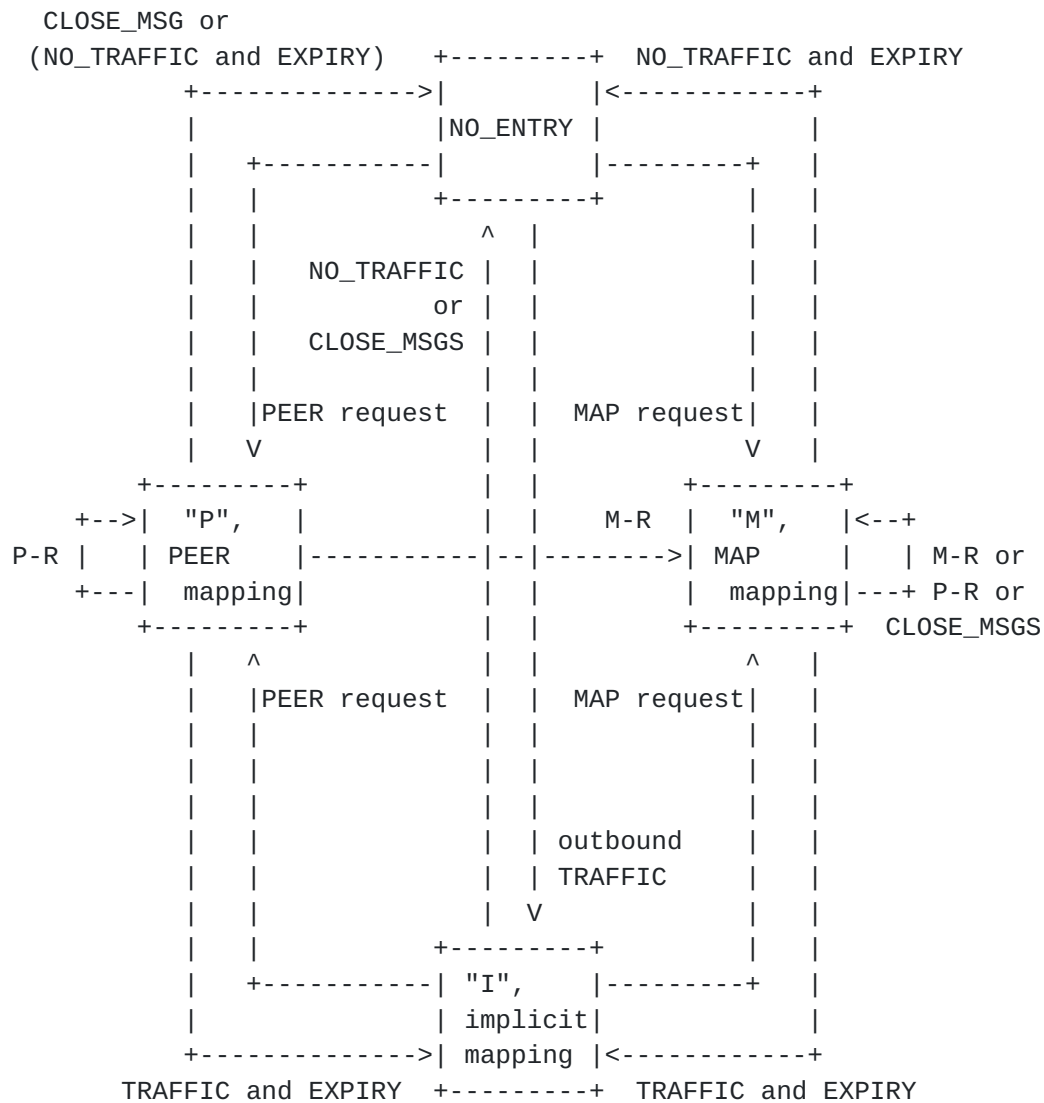


Figure 16: PCP State Diagram

The meanings of the states and events are:

NO_ENTRY: Invalid state represents Entry does not exist. This is the only possible start state.

M-R: MAP request

P-R: PEER request

M: Mapping entry when created by MAP request

P: Mapping entry when created/managed by PEER request

I: Implicit mapping created by an outgoing packet from the client (e.g., TCP SYN), and also the state when a PCP-created mapping's lifetime expires while there is still active traffic.

EXPIRY: PEER or MAP lifetime expired

TRAFFIC: Traffic seen by PCP-controlled device using this entry within the expiry time for that entry. This traffic may be inbound or outbound.

NO_TRAFFIC: Indicates that there is no TRAFFIC.

CLOSE_MSG: Protocol messages from the client or server to close the session (e.g., TCP FIN or TCP RST), as per the NAT or firewall device's handling of such protocol messages.

Notes on the diagram:

1. The 'and' clause indicates the events on either side of 'and' are required for the state-transition. The 'or' clause indicates either one of the events are enough for the state-transition.
2. Transition from state M to state I is implementation dependent.

15. Deployment Considerations

15.1. Ingress Filtering

As with implicit dynamic mappings created by outgoing TCP packets, explicit dynamic mappings created via PCP use the source IP address of the packet as the Internal Address for the mappings. Therefore ingress filtering [[RFC2827](#)] should be used on the path between the Internal Host and the PCP Server to prevent the injection of spoofed

packets onto that path.

15.2. Mapping Quota

On PCP-controlled devices that create state when a mapping is created (e.g., NAT), the PCP server SHOULD maintain per-host and/or per-subscriber quotas for mappings. It is implementation-specific whether the PCP server uses a separate quotas for implicit, explicit, and static mappings, a combined quota for all of them, or some other policy.

16. Security Considerations

The goal of the PCP protocol is to improve the ability of end nodes to control their associated NAT state, and to improve the efficiency and error handling of NAT mappings when compared to existing implicit mapping mechanisms in NAT boxes and stateful firewalls. It is the security goal of the PCP protocol to limit any new denial of service opportunities, and to avoid introducing new attacks that can result in unauthorized changes to mapping state. One of the most serious consequences of unauthorized changes in mapping state is traffic theft. All mappings that could be created by a specific host using implicit mapping mechanisms are inherently considered to be authorized. Confidentiality of mappings is not a requirement, even in cases where the PCP messages may transit paths that would not be travelled by the mapped traffic.

16.1. Simple Threat Model

PCP is secure against off-path attackers who cannot spoof a packet that the PCP Server will view as a packet received from the internal network.

Defending against attackers who can modify or drop packets between the internal network and the PCP server, or who can inject spoofed packets that appear to come from the internal network is out of scope.

A PCP Server is secure under this threat model if the PCP Server is constrained so that it does not configure any explicit mapping that it would not configure implicitly. In most cases, this means that PCP Servers running on NAT boxes or stateful firewalls that support the PEER Opcode can be secure under this threat model if all of their hosts are within a single administrative domain (or if the internal hosts can be securely partitioned into separate administrative domains, as in the DS-Lite B4 case), explicit mappings are created with the same lifetime as implicit mappings, the PCP server does not

support deleting or reducing the lifetime of existing mappings, and the PCP server does not support the third party option. PCP Servers can also securely support the MAP Opcode under this threat model if the security policy on the device running the PCP Server would permit endpoint independent filtering of implicit mappings.

PCP Servers that comply with the Simple Threat Model and do not implement a PCP security mechanism described in [Section 16.2](#) MUST enforce the constraints described in the paragraph above.

16.1.1. Attacks Considered

- o If you allow multiple administrative domains to send PCP requests to a single PCP server that does not enforce a boundary between the domains, it is possible for a node in one domain to perform a denial of service attack on other domains, or to capture traffic that is intended for a node in another domain.
- o If explicit mappings have longer lifetimes than implicit mappings, it makes it easier to perpetrate a denial of service attack than it would be if the PCP Server was not present.
- o If the PCP Server supports deleting or reducing the lifetime of existing mappings, this allows an attacking node to steal an existing mapping and receive traffic that was intended for another node.
- o If the THIRD_PARTY Option is supported, this also allows an attacker to open a window for an external node to attack an internal node, allows an attacker to steal traffic that was intended for another node, or may facilitate a denial of service attack. One example of how the THIRD_PARTY Option could grant an attacker more capability than a spoofed implicit mapping is that the PCP server (especially if it is running in a service provider's network) may not be aware of internal filtering that would prevent spoofing an equivalent implicit mapping, such as filtering between a guest and corporate network.
- o If the MAP Opcode is supported by the PCP server in cases where the security policy would not support endpoint independent filtering of implicit mappings, then the MAP Opcode changes the security properties of the device running the PCP Server by allowing explicit mappings that violate the security policy.

16.1.2. Deployment Examples Supporting the Simple Threat Model

This section offers two examples of how the Simple Threat Model can be supported in real-world deployment scenarios.

16.1.2.1. Residential Gateway Deployment

Parity with many currently-deployed residential gateways can be achieved using a PCP Server that is constrained as described in [Section 16.1.1](#) above.

16.1.2.2. DS-Lite Deployment

A DS-Lite deployment could be secure under the Simple Threat Model, even if the B4 device makes PCP mapping requests on behalf of internal clients using the THIRD_PARTY option. In this case the DS-Lite PCP server MUST be configured to only allow the B4 device to make THIRD_PARTY requests, and only on behalf of other Internal Hosts sharing the same DS-Lite IPv6 tunnel. The B4 device MUST guard against spoofed packets being injected into the IPv6 tunnel using the B4 device's IPv4 source address, so the DS-Lite PCP Server can trust that packets received over the DS-Lite IPv6 tunnel with the B4 device's source IPv4 address do in fact originate from the B4 device. The B4 device is in a position to enforce this requirement, because it is the DS-Lite IPv6 tunnel endpoint.

Allowing the B4 device to use the THIRD_PARTY Option to create mappings for hosts reached via the IPv6 tunnel terminated by the B4 device is acceptable, because the B4 device is capable of creating these mappings implicitly and can prevent others from spoofing these mappings.

16.2. Advanced Threat Model

In the Advanced Threat Model the PCP protocol must ensure that attackers (on- or off-path) cannot create unauthorized mappings or make unauthorized changes to existing mappings. The protocol must also limit the opportunity for on- or off-path attackers to perpetrate denial of service attacks.

The Advanced Threat Model security model will be needed in the following cases:

- o Security infrastructure equipment, such as corporate firewalls, that does not create implicit mappings.
- o Equipment (such as CGNs or service provider firewalls) that serve multiple administrative domains and do not have a mechanism to

securely partition traffic from those domains.

- o Any implementation that wants to be more permissive in authorizing explicit mappings than it is in authorizing implicit mappings.
- o Implementations that support the THIRD_PARTY Option (unless they can meet the constraints outlined in [Section 16.1.2.2](#)).
- o Implementations that wish to support any deployment scenario that does not meet the constraints described in [Section 16.1](#).

To protect against attacks under this threat model, a PCP security mechanism which provides an authenticated, integrity protected signaling channel would need to be specified.

PCP Servers that implement a PCP security mechanism MAY accept unauthenticated requests. PCP Servers implementing the PCP security mechanism MUST enforce the constraints described in [Section 16.1](#) above, in their default configuration, when processing unauthenticated requests.

[16.3](#). Residual Threats

This section describes some threats that are not addressed in either of the above threat models, and recommends appropriate mitigation strategies.

[16.3.1](#). Denial of Service

Because of the state created in a NAT or firewall, a per-host and/or per-subscriber quota will likely exist for both implicit dynamic mappings and explicit dynamic mappings. A host might make an excessive number of implicit or explicit dynamic mappings, consuming an inordinate number of ports, causing a denial of service to other hosts. Thus, [Section 15.2](#) recommends that hosts be limited to a reasonable number of explicit dynamic mappings.

An attacker, on the path between the PCP client and PCP server, can drop PCP requests, drop PCP responses, or spoof a PCP error, all of which will effectively deny service. Through such actions, the PCP client might not be aware the PCP server might have actually processed the PCP request. An attacker sending a NO_RESOURCES error can cause the PCP client to not send messages to that server for a while.

16.3.2. Ingress Filtering

It is important to prevent a host from fraudulently creating, deleting, or refreshing a mapping (or filtering) for another host, because this can expose the other host to unwanted traffic, prevent it from receiving wanted traffic, or consume the other host's mapping quota. Both implicit and explicit dynamic mappings are created based on the source IP address in the packet, and hence depend on ingress filtering to guard against spoof source IP addresses.

16.3.3. Mapping Theft

In the time between when a PCP server loses state and the PCP client notices the lower than expected Epoch Time value, it is possible that the PCP client's mapping will be acquired by another host (via an explicit dynamic mapping or implicit dynamic mapping). This means incoming traffic will be sent to a different host ("theft"). Rapid Recovery reduces this interval, but would not completely eliminate this threat. The PCP client can reduce this interval by using a relatively short lifetime; however, this increases the amount of PCP chatter. This threat is reduced by using persistent storage of explicit dynamic mappings in the PCP server (so it does not lose explicit dynamic mapping state), or by ensuring the previous external IP address and port cannot be used by another host (e.g., by using a different IP address pool).

16.3.4. Attacks Against Server Discovery

This document does not specify server discovery, beyond contacting the default gateway.

17. IANA Considerations

IANA is requested to perform the following actions:

17.1. Port Number

PCP will use ports 5350 and 5351 (currently assigned by IANA to NAT-PMP [[I-D.cheshire-nat-pmp](#)]). We request that IANA re-assign those ports to PCP, and relinquish UDP port 44323.

[Note to RFC Editor: Please remove the text about relinquishing port 44323 prior to publication.]

17.2. Opcodes

IANA shall create a new protocol registry for PCP Opcodes, numbered 0-127, initially populated with the values:

value	Opcode
-----	-----
0	ANNOUNCE
1	MAP
2	PEER
3-95	(specification required)
96-126	(private use)
127	Reserved

The value 127 is Reserved and may be assigned via Standards Action [[RFC5226](#)]. The values in the range 3-95 can be assigned via Specification Required [[RFC5226](#)], and the range 96-126 is for Private Use [[RFC5226](#)].

17.3. Result Codes

IANA shall create a new registry for PCP result codes, numbered 0-255, initially populated with the result codes from [Section 6.4](#). The value 255 is Reserved and may be assigned via Standards Action [[RFC5226](#)].

Result Codes in the range 13-191 can be assigned via Specification Required [[RFC5226](#)], and the range 192-254 is for Private Use [[RFC5226](#)].

17.4. Options

IANA shall create a new registry for PCP Options, numbered 0-255 with an associated mnemonic. The values 0-127 are mandatory-to-process, and 128-255 are optional to process. The initial registry contains the Options described in [Section 12](#). The Option values 0, 127 and 255 are Reserved and may be assigned via Standards Action [[RFC5226](#)].

Additional PCP Option codes in the ranges 4-63 and 128-191 can be created via Specification Required [[RFC5226](#)], and the ranges 64-126 and 192-254 are for Private Use [[RFC5226](#)].

18. Acknowledgments

Thanks to Xiaohong Deng, Alain Durand, Christian Jacquenet, Jacni Qin, Simon Perreault, James Yu, Tina TSOU (Ting ZOU), Felipe Miranda Costa, James Woodyatt, Dave Thaler, and Masataka Ohta for their

comments and review. Thanks to Simon Perreault for highlighting the interaction of dynamic connections with PCP-created mappings.

Thanks to Francis Dupont for his several thorough reviews of the specification, which improved the protocol significantly.

Thanks to T. S. Ranganathan for the state diagram.

Thanks to Margaret Wasserman for writing the Security Considerations section.

19. References

19.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, [RFC 768](#), August 1980.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", [RFC 2136](#), April 1997.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", [BCP 38](#), [RFC 2827](#), May 2000.
- [RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", [RFC 3007](#), November 2000.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", [RFC 4193](#), October 2005.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 5226](#), May 2008.
- [RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", [BCP 156](#), [RFC 6056](#), January 2011.
- [proto_numbers] IANA, "Protocol Numbers", 2011, <<http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml>>.

19.2. Informative References

- [Bonjour] "Bonjour",
<[http://en.wikipedia.org/wiki/Bonjour_\(software\)](http://en.wikipedia.org/wiki/Bonjour_(software))>.
- [I-D.arkko-dual-stack-extra-lite]
Arkko, J., Eggert, L., and M. Townsley, "Scalable Operation of Address Translators with Per-Interface Bindings", [draft-arkko-dual-stack-extra-lite-05](#) (work in progress), February 2011.
- [I-D.boucadair-pcp-failure]
Boucadair, M., Dupont, F., and R. Penno, "Port Control Protocol (PCP) Failure Scenarios", [draft-boucadair-pcp-failure-02](#) (work in progress), September 2011.
- [I-D.bpw-pcp-upnp-igd-interworking]
Boucadair, M., Penno, R., Wing, D., and F. Dupont, "Universal Plug and Play (UPnP) Internet Gateway Device (IGD)-Port Control Protocol (PCP) Interworking Function", [draft-bpw-pcp-upnp-igd-interworking-02](#) (work in progress), February 2011.
- [I-D.cheshire-dnsext-dns-sd]
Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", [draft-cheshire-dnsext-dns-sd-11](#) (work in progress), December 2011.
- [I-D.cheshire-nat-pmp]
Cheshire, S., "NAT Port Mapping Protocol (NAT-PMP)", [draft-cheshire-nat-pmp-03](#) (work in progress), April 2008.
- [I-D.ietf-behave-lsn-requirements]
Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common requirements for Carrier Grade NATs (CGNs)", [draft-ietf-behave-lsn-requirements-05](#) (work in progress), November 2011.
- [I-D.ietf-behave-sctpnat]
Stewart, R., Tuexen, M., and I. Ruengeler, "Stream Control Transmission Protocol (SCTP) Network Address Translation", [draft-ietf-behave-sctpnat-05](#) (work in progress), June 2011.
- [I-D.miles-behave-l2nat]
Miles, D. and M. Townsley, "Layer2-Aware NAT", [draft-miles-behave-l2nat-00](#) (work in progress),

March 2009.

- [IGDv1] UPnP Gateway Committee, "WANIPConnection:1", November 2001, <<http://upnp.org/specs/gw/UPnP-gw-WANIPConnection-v1-Service.pdf>>.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, [RFC 793](#), September 1981.
- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", [BCP 5](#), [RFC 1918](#), February 1996.
- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", [RFC 3022](#), January 2001.
- [RFC3581] Rosenberg, J. and H. Schulzrinne, "An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing", [RFC 3581](#), August 2003.
- [RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global Unicast Address Format", [RFC 3587](#), August 2003.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", [RFC 4291](#), February 2006.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", [RFC 4340](#), March 2006.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", [BCP 127](#), [RFC 4787](#), January 2007.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", [RFC 4941](#), September 2007.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", [RFC 4960](#), September 2007.
- [RFC4961] Wing, D., "Symmetric RTP / RTP Control Protocol (RTCP)", [BCP 131](#), [RFC 4961](#), July 2007.
- [RFC5382] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", [BCP 142](#), [RFC 5382](#), October 2008.

- [RFC6092] Woodyatt, J., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", [RFC 6092](#), January 2011.
- [RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", [RFC 6145](#), April 2011.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", [RFC 6146](#), April 2011.
- [RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", [RFC 6296](#), June 2011.
- [RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", [RFC 6333](#), August 2011.

[Appendix A.](#) NAT-PMP Transition

The Port Control Protocol (PCP) is a successor to the NAT Port Mapping Protocol, NAT-PMP [[I-D.cheshire-nat-pmp](#)], and shares similar semantics, concepts, and packet formats. Because of this NAT-PMP and PCP both use the same port, and use NAT-PMP and PCP's version negotiation capabilities to determine which version to use. This section describes how an orderly transition may be achieved.

A client supporting both NAT-PMP and PCP SHOULD send its request using the PCP packet format. This will be received by a NAT-PMP server or a PCP server. If received by a NAT-PMP server, the response will be as indicated by the NAT-PMP specification [[I-D.cheshire-nat-pmp](#)], which will cause the client to downgrade to NAT-PMP and re-send its request in NAT-PMP format. If received by a PCP server, the response will be as described by this document and processing continues as expected.

A PCP server supporting both NAT-PMP and PCP can handle requests in either format. The first octet of the packet indicates if it is NAT-PMP (first octet zero) or PCP (first octet non-zero).

A PCP-only gateway receiving a NAT-PMP request (identified by the first octet being zero) will interpret the request as a version mismatch. Normal PCP processing will emit a PCP response that is compatible with NAT-PMP, without any special handling by the PCP server.

Appendix B. Change History

[Note to RFC Editor: Please remove this section prior to publication.]

B.1. Changes from [draft-ietf-pcp-base-22](#) to -23

- o Instead of returning error NO_RESOURCES when requesting a MAP for all protocols or for all ports, return UNSUPP_PROTOCOL.
- o Clarify that PEER-created mappings are treated as if it was implicit dynamic outbound mapping ([Section 11.3](#)).
- o Point out that PEER-created mappings may be very short until bi-directional traffic is seen by the PCP-managed device.
- o Clarification that an existing implicit mapping (created e.g., by TCP SYN) can become managed by a MAP request ([Section 10.3](#)).
- o Clarified the ANNOUNCE Opcode is being defined in [Section 13.1](#), and that the length of requests (as well as responses) is zero.
- o Clarify that ANNOUNCE has Lifetime=0 for requests and responses.
- o Clarify ANNOUNCE can be sent unicast by the client (to solicit a response), or can be multicasted (unsolicited) by the server.
- o Allow ANNOUNCE to be sent unicast by the server, to accomodate case where PCP server fails but knows the IP address of a PCP client (e.g., web portal).
- o Clarified ports used for unicast and multicast unsolicited ANNOUNCE.
- o Tweaked NO_RESOURCES handling, to just disallow *new* mappings.
- o State diagram is now non-normative, because it overly simplifies that implicit mappings become MAP (when they actually still retain their previous behavior when the MAP expires).
- o In section [Section 10.5](#), clarified that PEER cannot delete or shorten any lifetime, and that MAP can only shorten or delete lifetimes of MAP-created mappings.
- o Clarified handling of MAP when mapping already exists (4 steps).
- o 2^32-1

- o Randomize retry interval (1.5-2.5), and maximum retry interval is now 1024 seconds (was 15 minutes).
- o Remove MUST be 0 for Reserved field when sending error responses for un-parseable message.
- o Whenever PCP client includes Suggested IP Address (in MAP or PEER), the PCP server should try to fulfill that request, even if creating a mapping on that IP address means the internal host will have mappings on different IP addresses and ports.
- o For NO_RESOURCES error, the PCP client can attempt to renew and attempt to delete mappings (as they can help shed load) -- it just can't try to create new ones.
- o Removed the overly simplistic normative text regarding honoring Suggested External Address from [Section 9](#) in favor of the text in [Section 10.3](#) which has significantly more detail.

B.2. Changes from [draft-ietf-pcp-base-21](#) to -22

- o Removed paragraph discussing multiple addresses on the same (physical) interface; those will work with PCP.
- o The FILTER Option's Prefix Length field redefined to simply be a count of the relevant bits (rather than 0-32 for IPv4-mapped addresses).
- o Point out NO_RESOURCES attack vector in security considerations.
- o Tighten up recommendation for client handling long Lifetimes, and moved from the MAP-specific section to the General PCP Processing section. Client should normalize to 24 hours maximum for success and 30 minute maximum for errors.

B.3. Changes from [draft-ietf-pcp-base-20](#) to -21

- o To delete all mappings using THIRD_PARTY, use the all-zeros IP address (rather than previous text which used length=0).
- o added normative text for what PCP server does when it receives all-zeros IP address in THIRD_PARTY option.
- o PREFER_FAILURE allowed for use by web portal.
- o clarifications to mandatory option processing.

- o cleanup and wordsmithing of the THIRD_PARTY text.

B.4. Changes from [draft-ietf-pcp-base-19](#) to -20

- o clarify if Options are included in responses.
- o clarify when External Address can be ignored by the PCP server / PCP-controlled device
- o added 'Transition from state M to state I is implementation dependent' to state diagram

B.5. Changes from [draft-ietf-pcp-base-18](#) to -19

- o Described race condition with MAP containing PREFER_FAILURE and Mapping Update.
- o Added state machine ([Section 14.5](#)).
- o Fully integrated Rapid Recovery, with a separate Opcode having its own processing description.
- o Clarified that due to Mapping Update, a single MAP or PEER request can receive multiple responses, each updating the previous request, and that the PCP client needs to handle MAP updates or PEER updates accordingly.

B.6. Changes from [draft-ietf-pcp-base-17](#) to -18

- o Removed UNPROCESSED option. Instead, unprocessed options are simply not included in responses.
- o Updated terminology section for Implicit/Explicit and Outbound/Inbound.
- o PEER requests cannot delete or shorten the lifetime of a mapping.
- o Clarified that PCP clients only retransmit mapping requests for as long as they actually want the mapping.
- o Revised Epoch time calculations and explanation.
- o Renamed the announcement opcode from No-Op to ANNOUNCE.

B.7. Changes from [draft-ietf-pcp-base-16](#) to -17

- o suggest acquiring a mapping to the Discard port if there is a desire to show the user their external address ([Section 10.7](#)).
- o Added Restart Announcement.
- o Tweaked terminology.
- o Detailed how error responses are generated.

B.8. Changes from [draft-ietf-pcp-base-15](#) to -16

- o fixed mistake in PCP request format (had 32 bits of extraneous fields)
- o Allow MAP to request all ports (port=0) for a specific protocol (protocol!=0), for the same reason we added support for all ports (port=0) and all protocols (protocol=0) in -15
- o corrected text on Client Processing a Response related to receiving ADDRESS_MISMATCH error.
- o updated Epoch text.
- o Added text that MALFORMED_REQUEST is generated for MAP if Protocol is zero but Internal Port is non-zero.

B.9. Changes from [draft-ietf-pcp-base-14](#) to -15

- o Softened and removed text that was normatively explaining how PEER is implemented within a NAT.
- o Allow a MAP request for protocol=0, which means "all protocols". This can work for an IPv6 or IPv4 firewall. Its use with a NAPT is undefined.
- o combined SERVER_OVERLOADED and NO_RESOURCES into one error code, NO_RESOURCES.
- o SCTP mappings have to use same internal and suggested external ports, and have implied PREFER_FAILURE semantics.
- o Re-instated ADDRESS_MISMATCH error, which only checks the client address (not its port).

B.10. Changes from [draft-ietf-pcp-base-13](#) to -14

- o Moved discussion of socket operations for PCP source address into Implementation Considerations section.
- o Integrated numerous WGLC comments.
- o NPTv6 in scope.
- o Re-written security considerations section. Thanks, Margaret!
- o Reduced PEER4 and PEER6 Opcodes to just a single Opcode, PEER.
- o Reduced MAP4 and MAP6 Opcodes to just a single Opcode, MAP.
- o Rearranged the PEER packet formats to align with MAP.
- o Removed discussion of the "O" bit for Options, which was confusing. Now the text just discusses the most significant bit of the Option code which indicates mandatory/optional, so it is clearer the field is 8 bits.
- o The THIRD_PARTY Option from an unauthorized host generates UNSUPP_OPTION, so the PCP server doesn't disclose it knows how to process THIRD_PARTY Option.
- o Added table to show which fields of MAP or PEER need IPv6/IPv4 addresses for IPv4 firewall, DS-Lite, NAT64, NAT44, etc.
- o Accommodate the server's Epoch going up or down, to better detect switching to a different PCP server.
- o Removed ADDRESS_MISMATCH; the server always includes its idea of the Client's IP Address and Port, and it's up to the client to detect a mismatch (and rectify it).

B.11. Changes from [draft-ietf-pcp-base-12](#) to -13

- o All addresses are 128 bits. IPv4 addresses are represented by IPv4-mapped IPv6 addresses (::FFFF/96)
- o PCP request header now includes PCP client's port (in addition to the client's IP address, which was in -12).
- o new ADDRESS_MISMATCH error.
- o removed PROCESSING_ERROR error, which was too similar to MALFORMED_REQUEST.

- o Tweaked text describing how PCP client deals with multiple PCP server addresses ([Section 7.1](#))
- o clarified that when overloaded, the server can send SERVER_OVERLOADED (and drop requests) or simply drop requests.
- o Clarified how PCP client chooses MAP4 or MAP6, depending on the presence of its own IPv6 or IPv4 interfaces ([Section 9](#)).
- o compliant PCP server MUST support MAPx and PEERx, SHOULD support ability to disable support.
- o clarified that MAP-created mappings have no filtering, and PEER-created mappings have whatever filtering and mapping behavior is normal for that particular NAT / firewall.
- o Integrated WGLC feedback (small changes to abstract, definitions, and small edits throughout the document)
- o allow new Options to be defined with a specification (rather than standards action)

B.12. Changes from [draft-ietf-pcp-base-11](#) to -12

- o added implementation note that MAP and implicit dynamic mappings have independent mapping lifetimes.

B.13. Changes from [draft-ietf-pcp-base-10](#) to -11

- o clarified what can cause CANNOT_PROVIDE_EXTERNAL error to be generated.

B.14. Changes from [draft-ietf-pcp-base-09](#) to -10

- o Added External_AF field to PEER requests. Made PEER's Suggested External IP Address and Assigned External IP Address always be 128 bits long.

B.15. Changes from [draft-ietf-pcp-base-08](#) to -09

- o Clarified in PEER Opcode introduction ([Section 11](#)) that they can also create mappings.
- o More clearly explained how PEER can re-create an implicit dynamic mapping, for purposes of rebuilding state to maintain an existing session (e.g., long-lived TCP connection to a server).

- o Added Suggested External IP Address to the PEER Opcodes, to allow more robust rebuilding of connections. Added related text to the PEER server processing section.
- o Removed text encouraging PCP server to statefully remember its mappings from [Section 14.3.1](#), as it didn't belong there. Text in Security Considerations already encourages persistent storage.
- o More clearly discussed how PEER is used to re-establish TCP mapping state. Moved it to a new section, as well (it is now [Section 9.4](#)).
- o MAP errors now copy the Suggested Address (and port) fields to Assigned IP Address (and port), to allow PCP client to distinguish among many outstanding requests when using PREFER_FAILURE.
- o Mapping theft can also be mitigated by ensuring hosts can't re-use same IP address or port after state loss.
- o the UNPROCESSED option is renumbered to 0 (zero), which ensures no other option will be given 0 and be unable to be expressed by the UNPROCESSED option (due to its 0 padding).
- o created new Implementation Considerations section ([Section 14](#)) which discusses non-normative things that might be useful to implementers. Some new text is in here, and the Failure Scenarios text ([Section 14.3](#)) has been moved to here.
- o Tweaked wording of EDM NATs in [Section 14.1](#) to clarify the problem occurs both inside->outside and outside->inside.
- o removed "Interference by Other Applications on Same Host" section from security considerations.
- o fixed zero/non-zero text in [Section 10.5](#).
- o removed duplicate text saying MAP is allowed to delete an implicit dynamic mapping. It is still allowed to do that, but it didn't need to be said twice in the same paragraph.
- o Renamed error from UNAUTH_TARGET_ADDRESS to UNAUTH_THIRD_PARTY_INTERNAL_ADDRESS.
- o for FILTER option, removed unnecessary detail on how FILTER would be bad for PEER, as it is only allowed for MAP anyway.
- o In Security Considerations, explain that PEER can create a mapping which makes its security considerations the same as MAP.

B.16. Changes from [draft-ietf-pcp-base-07](#) to -08

- o moved all MAP4-, MAP6-, and PEER-specific options into a single section.
- o discussed NAT port-overloading and its impact on MAP (new section [Section 14.1](#)), which allowed removing the IMPLICIT_MAPPING_EXISTS error.
- o eliminated NONEXIST_PEER error (which was returned if a PEER request was received without an implicit dynamic mapping already being created), and adjusted PEER so that it creates an implicit dynamic mapping.
- o Removed Deployment Scenarios section (which detailed NAT64, NAT44, Dual-Stack Lite, etc.).
- o Added Client's IP Address to PCP common header. This allows server to refuse a PCP request if there is a mismatch with the source IP address, such as when a non-PCP-aware NAT was on the path. This should reduce failure situations where PCP is deployed in conjunction with a non-PCP-aware NAT. This addition was consensus at IETF80.
- o Changed UNSPECIFIED_ERROR to PROCESSING_ERROR. Clarified that MALFORMED_REQUEST is for malformed requests (and not related to failed attempts to process the request).
- o Removed MISORDERED_OPTIONS. Consensus of IETF80.
- o SERVER_OVERLOADED is now a common PCP error (instead of specific to MAP).
- o Tweaked PCP retransmit/retry algorithm again, to allow more aggressive PCP discovery if an implementation wants to do that.
- o Version negotiation text tweaked to soften NAT-PMP reference, and more clearly explain exactly what UNSUPP_VERSION should return.
- o PCP now uses NAT-PMP's UDP port, 5351. There are no normative changes to NAT-PMP or PCP to allow them both to use the same port number.
- o New [Appendix A](#) to discuss NAT-PMP / PCP interworking.
- o improved pseudocode to be non-blocking.

- o clarified that PCP cannot delete a static mapping (i.e., a mapping created by CLI or other non-PCP means).
- o moved theft of mapping discussion from Epoch section to Security Considerations.

B.17. Changes from [draft-ietf-pcp-base-06](#) to -07

- o tightened up THIRD_PARTY security discussion. Removed "highest numbered address", and left it as simply "the CPE's IP address".
- o removed UNABLE_TO_DELETE_ALL error.
- o renumbered Opcodes
- o renumbered some error codes
- o assigned value to IMPLICIT_MAPPING_EXISTS.
- o UNPROCESSED can include arbitrary number of option codes.
- o Moved lifetime fields into common request/response headers
- o We've noticed we're having to repeatedly explain to people that the "requested port" is merely a hint, and the NAT gateway is free to ignore it. Changed name to "suggested port" to better convey this intention.
- o Added NAT-PMP transition section
- o Separated Internal Address, External Address, Remote Peer Address definition
- o Unified Mapping, Port Mapping, Port Forwarding definition
- o adjusted so DHCP configuration is non-normative.
- o mentioned PCP refreshes need to be sent over the same interface.
- o renamed the REMOTE_PEER_FILTER option to FILTER.
- o Clarified FILTER option to allow sending an ICMP error if policy allows.
- o for MAP, clarified that if the PCP client changed its IP address and still wants to receive traffic, it needs to send a new MAP request.

- o clarified that PEER requests have to be sent from same interface as the connection itself.
- o for MAP opcode, text now requires mapping be deleted when lifetime expires (per consensus on 8-Mar interim meeting)
- o PEER Opcode: better description of remote peer's IP address, specifically that it does not control or establish any filtering, and explaining why it is 'from the PCP client's perspective'.
- o Removed latent text allowing DMZ for 'all protocols' (protocol=0). Which wouldn't have been legal, anyway, as protocol 0 is assigned by IANA to HOPOPT (thanks to James Yu for catching that one).
- o clarified that PCP server only listens on its internal interface.
- o abandoned 'target' term and reverted to simpler 'internal' term.

B.18. Changes from [draft-ietf-pcp-base-05](#) to -06

- o Dual-Stack Lite: consensus was encapsulation mode. Included a suggestion that the B4 will need to proxy PCP-to-PCP and UPnP-to-PCP.
- o defined THIRD_PARTY Option to work with the PEER Opcode, too. This meant moving it to its own section, and having both MAP and PEER Opcodes reference that common section.
- o used "target" instead of "internal", in the hopes that clarifies internal address used by PCP itself (for sending its packets) versus the address for MAPpings.
- o Options are now required to be ordered in requests, and ordering has to be validated by the server. Intent is to ease server processing of mandatory-to-implement options.
- o Swapped Option values for the mandatory- and optional-to-process Options, so we can have a simple lowest..highest ordering.
- o added MISORDERED_OPTIONS error.
- o re-ordered some error messages to cause MALFORMED_REQUEST (which is PCP's most general error response) to be error 1, instead of buried in the middle of the error numbers.
- o clarified that, after successfully using a PCP server, that PCP server is declared to be non-responsive after 5 failed retransmissions.

- o tightened up text (which was inaccurate) about how long general PCP processing is to delay when receiving an error and if it should honor Opcode-specific error lifetime. Useful for MAP errors which have an error lifetime. (This all feels awkward to have only some errors with a lifetime.)
- o Added better discussion of multiple interfaces, including highlighting WiFi+Ethernet. Added discussion of using IPv6 Privacy Addresses and [RFC1918](#) as source addresses for PCP requests. This should finish the section on multi-interface issues.
- o added some text about why server might send SERVER_OVERLOADED, or might simply discard packets.
- o Dis-allow internal-port=0, which means we dis-allow using PCP as a DMZ-like function. Instead, ports have to be mapped individually.
- o Text describing server's processing of PEER is tightened up.
- o Server's processing of PEER now says it is implementation-specific if a PCP server continues to allow the mapping to exist after a PEER message. Client's processing of PEER says that if client wants mapping to continue to exist, client has to continue to send recurring PEER messages.

B.19. Changes from [draft-ietf-pcp-base-04](#) to -05

- o tweaked PCP common header packet layout.
- o Re-added port=0 (all ports).
- o minimum size is 12 octets (missed that change in -04).
- o removed Lifetime from PCP common header.
- o for MAP error responses, the lifetime indicates how long the server wants the client to avoid retrying the request.
- o More clearly indicated which fields are filled by the server on success responses and error responses.
- o Removed UPnP interworking section from this document. It will appear in [[I-D.bpw-pcp-upnp-igd-interworking](#)].

B.20. Changes from [draft-ietf-pcp-base-03](#) to -04

- o "Pinhole" and "PIN" changed to "mapping" and "MAP".
- o Reduced from four MAP Opcodes to two. This was done by implicitly using the address family of the PCP message itself.
- o New option THIRD_PARTY, to more carefully split out the case where a mapping is created to a different host within the home.
- o Integrated a lot of editorial changes from Stuart and Francis.
- o Removed nested NAT text into another document, including the IANA-registered IP addresses for the PCP server.
- o Removed suggestion (MAY) that PCP server reserve UDP when it maps TCP. Nobody seems to need that.
- o Clearly added NAT and NAPT, such as in residential NATs, as within scope for PCP.
- o HONOR_EXTERNAL_PORT renamed to PREFER_FAILURE
- o Added 'Lifetime' field to the common PCP header, which replaces the functions of the 'temporary' and 'permanent' error types of the previous version.
- o Allow arbitrary Options to be included in PCP response, so that PCP server can indicate un-supported PCP Options. Satisfies PCP Issue #19
- o Reduced scope to only deal with mapping protocols that have port numbers.
- o Reduced scope to not support DMZ-style forwarding.
- o Clarified version negotiation.

B.21. Changes from [draft-ietf-pcp-base-02](#) to -03

- o Adjusted abstract and introduction to make it clear PCP is intended to forward ports and intended to reduce application keepalives.
- o First bit in PCP common header is set. This allows DTLS and non-DTLS to be multiplexed on same port, should a future update to this specification add DTLS support.

- o Moved subscriber identity from common PCP section to MAP* section.
- o made clearer that PCP client can reduce mapping lifetime if it wishes.
- o Added discussion of host running a server, client, or symmetric client+server.
- o Introduced PEER4 and PEER6 Opcodes.
- o Removed REMOTE_PEER Option, as its function has been replaced by the new PEER Opcodes.
- o IANA assigned port 44323 to PCP.
- o Removed AMBIGUOUS error code, which is no longer needed.

B.22. Changes from [draft-ietf-pcp-base-01](#) to -02

- o more error codes
- o PCP client source port number should be random
- o PCP message minimum 8 octets, maximum 1024 octets.
- o tweaked a lot of text in [section 7.4](#), "Opcode-Specific Server Operation".
- o opening a mapping also allows ICMP messages associated with that mapping.
- o PREFER_FAILURE value changed to the mandatory-to-process range.
- o added text recommending applications that are crashing obtain short lifetimes, to avoid consuming subscriber's port quota.

B.23. Changes from [draft-ietf-pcp-base-00](#) to -01

- o Significant document reorganization, primarily to split base PCP operation from Opcode operation.
- o packet format changed to move 'protocol' outside of PCP common header and into the MAP* opcodes
- o Renamed Informational Elements (IE) to Options.
- o Added REMOTE_PEER (for disambiguation with dynamic ports), REMOTE_PEER_FILTER (for simple packet filtering), and

PREFER_FAILURE (to optimize UPnP IGDv1 interworking) options.

- o Is NAT or router behind B4 in scope?
- o PCP option MAY be included in a request, in which case it MUST appear in a response. It MUST NOT appear in a response if it was not in the request.
- o Result code most significant bit now indicates permanent/temporary error
- o PCP Options are split into mandatory-to-process ("P" bit), and into Specification Required and Private Use.
- o Epoch discussion simplified.

Authors' Addresses

Dan Wing (editor)
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com

Stuart Cheshire
Apple Inc.
1 Infinite Loop
Cupertino, California 95014
USA

Phone: +1 408 974 3207
Email: cheshire@apple.com

Mohamed Boucadair
France Telecom
Rennes, 35000
France

Email: mohamed.boucadair@orange-ftgroup.com

Reinaldo Penno
Juniper Networks
1194 N Mathilda Avenue
Sunnyvale, California 94089
USA

Email: rpenno@juniper.net

Paul Selkirk
Internet Systems Consortium
950 Charter Street
Redwood City, California 94063
USA

Email: pselkirk@isc.org

