Network Working Group
Request for Comments: 2472
Obsoletes: 2023
Category: Standards Track

D. Haskin
E. Allen
Bay Networks, Inc.

December 1998

IP Version 6 over PPP

Status of this Memo

Copyright Notice

Abstract

The Point-to-Point Protocol (PPP) [1] provides a standard method of encapsulating Network Layer protocol information over point-to-point links. PPP also defines an extensible Link Control Protocol, and proposes a family of Network Control Protocols (NCPs) for establishing and configuring different network-layer protocols.

This document defines the method for transmission of IP Version 6 [2] packets over PPP links as well as the Network Control Protocol (NCP) for establishing and configuring the IPv6 over PPP. It also specifies the method of forming IPv6 link-local addresses on PPP links.

Table of Contents

1. Introduction

PPP has three main components:

1) A method for encapsulating datagrams over serial links.

2) A Link Control Protocol (LCP) for establishing, configuring, and testing the data-link connection.

3) A family of Network Control Protocols (NCPs) for establishing and configuring different network-layer protocols.

In order to establish communications over a point-to-point link, each end of the PPP link must first send LCP packets to configure and test the data link. After the link has been established and optional facilities have been negotiated as needed by the LCP, PPP must send NCP packets to choose and configure one or more network-layer protocols. Once each of the chosen network-layer protocols has been configured, datagrams from each network-layer protocol can be sent over the link.

In this document, the NCP for establishing and configuring the IPv6 over PPP is referred as the IPv6 Control Protocol (IPV6CP).

The link will remain configured for communications until explicit LCP or NCP packets close the link down, or until some external event occurs (power failure at the other end, carrier drop, etc.).

1.1. Specification of Requirements

In this document, several words are used to signify the requirements of the specification.

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 [7].

2. Sending IPv6 Datagrams

Before any IPv6 packets may be communicated, PPP MUST reach the Network-Layer Protocol phase, and the IPv6 Control Protocol MUST reach the Opened state.

Exactly one IPv6 packet is encapsulated in the Information field of PPP Data Link Layer frames where the Protocol field indicates type hex 0057 (Internet Protocol Version 6).

The maximum length of an IPv6 packet transmitted over a PPP link is the same as the maximum length of the Information field of a PPP data link layer frame. PPP links supporting IPv6 MUST allow the information field at least as large as the minimum link MTU size required for IPv6 [2].

3. A PPP Network Control Protocol for IPv6

The IPv6 Control Protocol (IPV6CP) is responsible for configuring, enabling, and disabling the IPv6 protocol modules on both ends of the point-to-point link. IPV6CP uses the same packet exchange mechanism as the Link Control Protocol (LCP). IPV6CP packets may not be exchanged until PPP has reached the Network-Layer Protocol phase. IPV6CP packets received before this phase is reached should be silently discarded.

The IPv6 Control Protocol is exactly the same as the Link Control Protocol [1] with the following exceptions:

Data Link Layer Protocol Field

Exactly one IPV6CP packet is encapsulated in the Information field of PPP Data Link Layer frames where the Protocol field indicates type hex 8057 (IPv6 Control Protocol).

Code field

Only Codes 1 through 7 (Configure-Request, Configure-Ack, Configure-Nak, Configure-Reject, Terminate-Request, Terminate-Ack and Code-Reject) are used. Other Codes should be treated as unrecognized and should result in Code-Rejects.

Timeouts

IPV6CP packets may not be exchanged until PPP has reached the Network-Layer Protocol phase. An implementation should be prepared to wait for Authentication and Link Quality Determination to finish before timing out waiting for a Configure-Ack or other response. It is suggested that an implementation give up only after user intervention or a configurable amount of time.

Configuration Option Types

IPV6CP has a distinct set of Configuration Options.

4. IPV6CP Configuration Options

IPV6CP Configuration Options allow negotiation of desirable IPv6 parameters. IPV6CP uses the same Configuration Option format defined for LCP [1], with a separate set of Options. If a Configuration Option is not included in a Configure-Request packet, the default value for that Configuration Option is assumed.

Up-to-date values of the IPV6CP Option Type field are specified in the most recent "Assigned Numbers" RFC [4]. Current values are assigned as follows:

```
1       Interface-Identifier
2       IPv6-Compression-Protocol
```

The only IPV6CP options defined in this document are Interface-Identifier and IPv6-Compression-Protocol. Any other IPV6CP configuration options that can be defined over time are to be defined in separate documents.

4.1. Interface-Identifier

Description

This Configuration Option provides a way to negotiate a unique 64-bit interface identifier to be used for the address autoconfiguration [3] at the local end of the link (see section 5). A Configure-Request MUST contain exactly one instance of the Interface-Identifier option [1]. The interface identifier MUST be unique within the PPP link; i.e. upon completion of the negotiation different Interface-Identifier values are to be selected for the ends of the PPP link. The interface identifier MAY also be unique over a broader scope.

Before this Configuration Option is requested, an implementation chooses its tentative Interface-Identifier. The non-zero value of the tentative Interface-Identifier SHOULD be chosen such that the value is both unique to the link and, if possible, consistently reproducible across initializations of the IPV6CP finite state machine (administrative Close and reOpen, reboots, etc). The rationale for preferring a consistently reproducible unique interface identifier to a completely random interface identifier is to provide stability to global scope addresses that can be formed from the interface identifier.

Assuming that interface identifier bits are numbered from 0 to 63 in canonical bit order where the most significant bit is the bit number 0, the bit number 6 is the "u" bit (universal/local bit

in IEEE EUI-64 [5] terminology) which indicates whether or not the interface identifier is based on a globally unique IEEE identifier (EUI-48 or EUI-64 [5]) (see the case 1 below). It is set to one (1) if a globally unique IEEE identifier is used to derive the interface identifier, and it is set to zero (0) otherwise.

The following are methods for choosing the tentative Interface Identifier in the preference order:

1) If an IEEE global identifier (EUI-48 or EUI-64) is available anywhere on the node, it should be used to construct the tentative Interface-Identifier due to its uniqueness properties. When extracting an IEEE global identifier from another device on the node, care should be taken to that the extracted identifier is presented in canonical ordering [8].

   The only transformation from an EUI-64 identifier is to invert the "u" bit (universal/local bit in IEEE EUI-64 terminology). For example, for a globally unique EUI-64 identifier of the form:

```
most-significant                                    least-significant
bit                                                              bit
|0              1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|cccccc0gccccccccc|ccccccccceeeeeee|eeeeeeeeeeeeeee|eeeeeeeeeeeeeee|
+----------------+----------------+----------------+----------------+
```

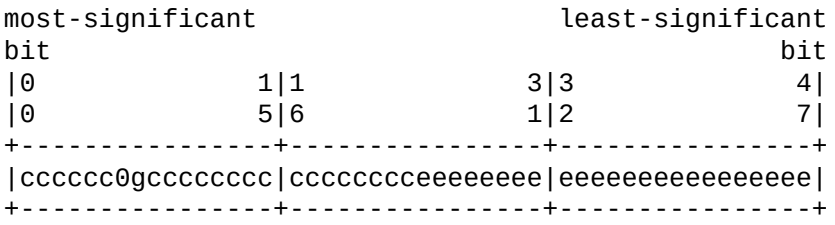
   where "c" are the bits of the assigned company_id, "0" is the value of the universal/local bit to indicate global scope, "g" is group/individual bit, and "e" are the bits of the extension identifier,

   the IPv6 interface identifier would be of the form:

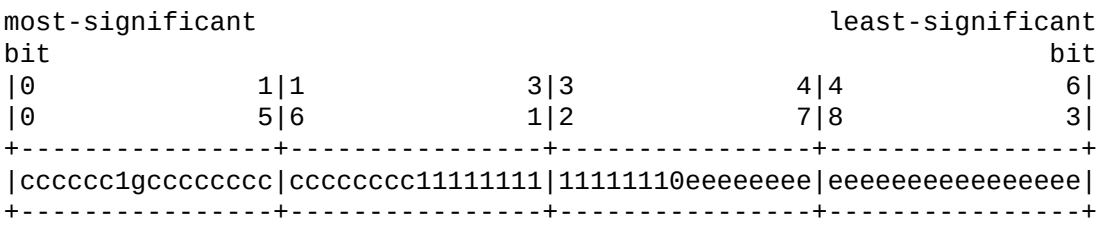
```
most-significant                                    least-significant
bit                                                              bit
|0              1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|cccccc1gccccccccc|ccccccccceeeeeee|eeeeeeeeeeeeeee|eeeeeeeeeeeeeee|
+----------------+----------------+----------------+----------------+
```

   The only change is inverting the value of the universal/local bit.

In the case of a EUI-48 identifier, it is first converted to the EUI-64 format by inserting two bytes, with hexadecimal values of 0xFF and 0xFE, in the middle of the 48 bit MAC (between the company_id and extension-identifier portions of the EUI-48 value). For example, for a globally unique 48 bit EUI-48 identifier of the form:



where "c" are the bits of the assigned company_id, "0" is the value of the universal/local bit to indicate global scope, "g" is group/individual bit, and "e" are the bits of the extension identifier, the IPv6 interface identifier would be of the form:



2) If an IEEE global identifier is not available a different source of uniqueness should be used. Suggested sources of uniqueness include link-layer addresses, machine serial numbers, et cetera.

   In this case the "u" bit of the interface identifier MUST be set to zero (0).

3) If a good source of uniqueness cannot be found, it is recommended that a random number be generated. In this case the "u" bit of the interface identifier MUST be set to zero (0).

Good sources [1] of uniqueness or randomness are required for the Interface-Identifier negotiation to succeed. If neither a unique number or a random number can be generated it is recommended that a zero value be used for the Interface-Identifier transmitted in the Configure-Request. In this case the PPP peer may provide a valid non-zero Interface-Identifier in its response as described below. Note that if at least one of the PPP peers is able to generate separate non-zero numbers for itself and its peer, the identifier negotiation will succeed.

When a Configure-Request is received with the Interface-Identifier Configuration Option and the receiving peer implements this option, the received Interface-Identifier is compared with the Interface-Identifier of the last Configure-Request sent to the peer. Depending on the result of the comparison an implementation MUST respond in one of the following ways:

If the two Interface-Identifiers are different but the received Interface-Identifier is zero, a Configure-Nak is sent with a non-zero Interface-Identifier value suggested for use by the remote peer. Such a suggested Interface-Identifier MUST be different from the Interface-Identifier of the last Configure-Request sent to the peer. It is recommended that the value suggested be consistently reproducible across initializations of the IPV6CP finite state machine (administrative Close and reOpen, reboots, etc). The "u" universal/local) bit of the suggested identifier MUST be set to zero (0) regardless of its source unless the globally unique EUI-48/EUI-64 derived identifier is provided for the exclusive use by the remote peer.

If the two Interface-Identifiers are different and the received Interface-Identifier is not zero, the Interface-Identifier MUST be acknowledged, i.e. a Configure-Ack is sent with the requested Interface-Identifier, meaning that the responding peer agrees with the Interface-Identifier requested.

If the two Interface-Identifiers are equal and are not zero, a Configure-Nak MUST be sent specifying a different non-zero Interface-Identifier value suggested for use by the remote peer. It is recommended that the value suggested be consistently reproducible across initializations of the IPV6CP finite state machine (administrative Close and reOpen, reboots, etc). The "u" universal/local) bit of the suggested identifier MUST be set to zero (0) regardless of its source unless the globally unique EUI-48/EUI-64 derived identifier is provided for the exclusive use by the remote peer.

If the two Interface-Identifiers are equal to zero, the Interface-Identifiers negotiation MUST be terminated by transmitting the Configure-Reject with the Interface-Identifier value set to zero. In this case a unique Interface-Identifier can not be negotiated.

If a Configure-Request is received with the Interface-Identifier Configuration Option and the receiving peer does not implement this option, Configure-Rej is sent.

A new Configure-Request SHOULD NOT be sent to the peer until normal processing would cause it to be sent (that is, until a Configure-Nak is received or the Restart timer runs out).

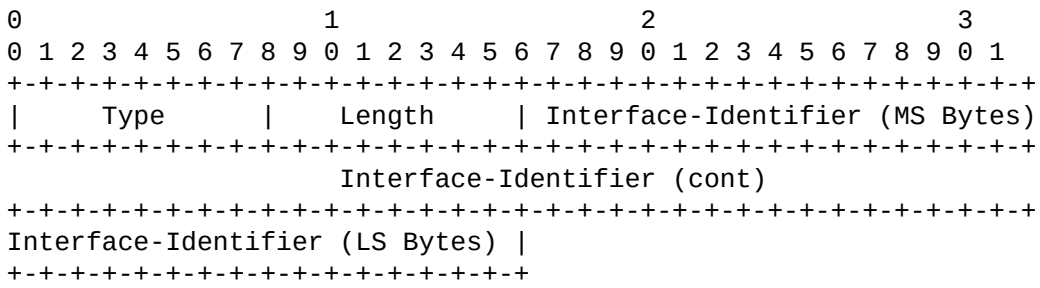
A new Configure-Request MUST NOT contain the Interface-Identifier option if a valid Interface-Identifier Configure-Reject is received.

Reception of a Configure-Nak with a suggested Interface-Identifier different from that of the last Configure-Nak sent to the peer indicates a unique Interface-Identifier. In this case a new Configure-Request MUST be sent with the identifier value suggested in the last Configure-Nak from the peer. But if the received Interface-Identifier is equal to the one sent in the last Configure-Nak, a new Interface-Identifier MUST be chosen. In this case, a new Configure-Request SHOULD be sent with the new tentative Interface-Identifier. This sequence (transmit Configure-Request, receive Configure-Request, transmit Configure-Nak, receive Configure-Nak) might occur a few times, but it is extremely unlikely to occur repeatedly. More likely, the Interface-Identifiers chosen at either end will quickly diverge, terminating the sequence.

If negotiation of the Interface-Identifier is required, and the peer did not provide the option in its Configure-Request, the option SHOULD be appended to a Configure-Nak. The tentative value of the Interface-Identifier given must be acceptable as the remote Interface-Identifier; i.e. it should be different from the identifier value selected for the local end of the PPP link. The next Configure-Request from the peer may include this option. If the next Configure-Request does not include this option the peer MUST NOT send another Configure-Nak with this option included. It should assume that the peer's implementation does not support this option.

By default, an implementation SHOULD attempt to negotiate the Interface-Identifier for its end of the PPP connection.

A summary of the Interface-Identifier Configuration Option format is shown below. The fields are transmitted from left to right.



Type

1

Length

10

Interface-Identifier

The 64-bit Interface-Identifier which is very likely to be unique on the link or zero if a good source of uniqueness can not be found.

Default

If no valid interface identifier can be successfully negotiated, no default Interface-Identifier value should be assumed. The procedures for recovering from such a case are unspecified. One approach is to manually configure the interface identifier of the interface.

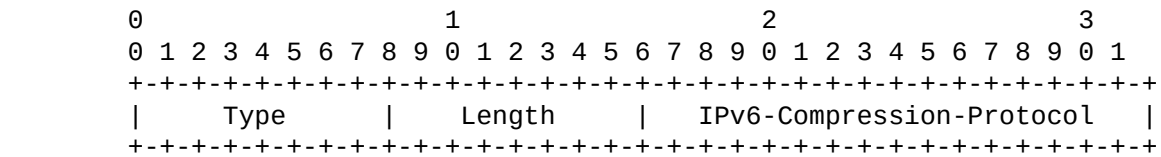
4.2. IPv6-Compression-Protocol

Description

This Configuration Option provides a way to negotiate the use of a specific IPv6 packet compression protocol. The IPv6-Compression-Protocol Configuration Option is used to indicate the ability to receive compressed packets. Each end of the link must separately request this option if bi-directional compression is desired. By default, compression is not enabled.

IPv6 compression negotiated with this option is specific to IPv6 datagrams and is not to be confused with compression resulting from negotiations via Compression Control Protocol (CCP), which potentially effect all datagrams.

A summary of the IPv6-Compression-Protocol Configuration Option format is shown below. The fields are transmitted from left to right.



Type

2

Length

>= 4

IPv6-Compression-Protocol

The IPv6-Compression-Protocol field is two octets and indicates the compression protocol desired. Values for this field are always the same as the PPP Data Link Layer Protocol field values for that same compression protocol.

No IPv6-Compression-Protocol field values are currently assigned. Specific assignments will be made in documents that define specific compression algorithms.

Data

The Data field is zero or more octets and contains additional data as determined by the particular compression protocol.

Default

No IPv6 compression protocol enabled.

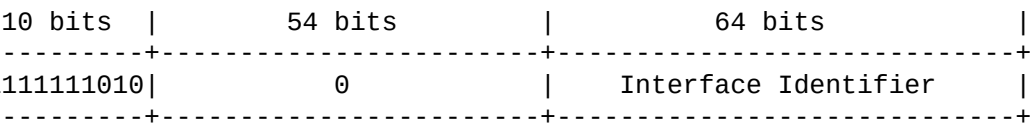
5. Stateless Autoconfiguration and Link-Local Addresses

The Interface Identifier of IPv6 unicast addresses [6] of a PPP interface, SHOULD be negotiated in the IPV6CP phase of the PPP connection setup (see section 4.1). If no valid Interface Identifier has been successfully negotiated, procedures for recovering from such a case are unspecified. One approach is to manually configure the Interface Identifier of the interface.

As long as the Interface Identifier is negotiated in the IPV6CP phase of the PPP connection setup, it is redundant to perform duplicate address detection as a part of the IPv6 Stateless Autoconfiguration

protocol [3]. Therefore it is recommended that for PPP links with the IPV6CP Interface-Identifier option enabled the default value of the DupAddrDetectTransmits autoconfiguration variable [3] be zero.

Link-local addresses of PPP interfaces have the following format:



The most significant 10 bits of the address is the Link-Local prefix FE80::. 54 zero bits pad out the address between the Link-Local prefix and the Interface Identifier fields.

6. Security Considerations

The IPv6 Control Protocol extension to PPP can be used with all defined PPP authentication and encryption mechanisms.

7. Acknowledgments

This document borrows from the Magic-Number LCP option and as such is partially based on previous work done by the PPP working group.

8. Changes from RFC-2023

The following changes were made from RFC-2023 "IP Version 6 over PPP":

- Changed to use "Interface Identifier" instead of the "Interface Token" term according to the terminology adopted in [6].

- Increased the size of Interface Identifier to 64 bits according to the newly adopted IPv6 addressing architecture [6].

- Added methods for selection of an interface identifier that is consistently reproducible across initializations of the IPV6CP finite state machine.

- Added the interface identifier selection methods for generating globally unique interface identifier from an unique an IEEE global identifier when it is available anywhere on the node.

- Changed to send a Configure-Nak instead a Configure-Ack in response to receiving a Configure-Request with a zero Interface-Identifier value.

- Replaced the value assignment of the IPv6-Compression-Protocol field of the IPv6-Compression-Protocol Configuration option with the text stating that no IPv6-Compression-Protocol field values are currently assigned and that specific assignments will be made in documents that define specific compression algorithms.

- Added new and updated references.

- Minor text clarifications and improvements.

9. References

[1] Simpson, W., "The Point-to-Point Protocol", STD 51, RFC 1661, July 1994.

[2] Deering, S., and R. Hinden, Editors, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[3] Thomson, S., and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[4] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994. See also: http://www.iana.org/numbers.html

[5] IEEE, "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority", http://standards.ieee.org/db/oui/tutorials/EUI64.html, March 1997.

[6] Hinden, R., and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, July 1998.

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

[8] Narten T., and C. Burton, "A Caution On The Canonical Ordering Of Link-Layer Addresses", RFC 2469, December 1998.

10. Authors' Addresses

Dimitry Haskin
Bay Networks, Inc.
600 Technology Park
Billerica, MA 01821

EMail: dhaskin@baynetworks.com

Ed Allen
Bay Networks, Inc.
600 Technology Park
Billerica, MA 01821

EMail: eallen@baynetworks.com

11. Full Copyright Statement