

Network Working Group  
Request for Comments: 2661  
Category: Standards Track

W. Townsley  
A. Valencia  
Cisco Systems  
A. Rubens  
Ascend Communications  
G. Pall  
G. Zorn  
Microsoft Corporation  
B. Palter  
Redback Networks  
August 1999

## Layer Two Tunneling Protocol "L2TP"

### Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

### Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

### Abstract

This document describes the Layer Two Tunneling Protocol (L2TP). STD 51, RFC 1661 specifies multi-protocol access via PPP [RFC1661]. L2TP facilitates the tunneling of PPP packets across an intervening network in a way that is as transparent as possible to both end-users and applications.

### Table of Contents

1.0 Introduction.....	3
1.1 Specification of Requirements.....	4
1.2 Terminology.....	4
2.0 Topology.....	8
3.0 Protocol Overview.....	9
3.1 L2TP Header Format.....	9
3.2 Control Message Types.....	11
4.0 Control Message Attribute Value Pairs.....	12
4.1 AVP Format.....	13
4.2 Mandatory AVPs.....	14
4.3 Hiding of AVP Attribute Values.....	14

4.4 AVP Summary.....	17
4.4.1 AVPs Applicable To All Control Messages.....	17
4.4.2 Result and Error Codes.....	18
4.4.3 Control Connection Management AVPs.....	20
4.4.4 Call Management AVPs.....	27
4.4.5 Proxy LCP and Authentication AVPs.....	34
4.4.6 Call Status AVPs.....	39
5.0 Protocol Operation.....	41
5.1 Control Connection Establishment.....	41
5.1.1 Tunnel Authentication.....	42
5.2 Session Establishment.....	42
5.2.1 Incoming Call Establishment.....	42
5.2.2 Outgoing Call Establishment.....	43
5.3 Forwarding PPP Frames.....	43
5.4 Using Sequence Numbers on the Data Channel.....	44
5.5 Keepalive (Hello).....	44
5.6 Session Teardown.....	45
5.7 Control Connection Teardown.....	45
5.8 Reliable Delivery of Control Messages.....	46
6.0 Control Connection Protocol Specification.....	48
6.1 Start-Control-Connection-Request (SCCRQ).....	48
6.2 Start-Control-Connection-Reply (SCCRP).....	48
6.3 Start-Control-Connection-Connected (SCCCN).....	49
6.4 Stop-Control-Connection-Notification (StopCCN).....	49
6.5 Hello (HELLO).....	49
6.6 Incoming-Call-Request (ICRQ).....	50
6.7 Incoming-Call-Reply (ICRP).....	51
6.8 Incoming-Call-Connected (ICCN).....	51
6.9 Outgoing-Call-Request (OCRQ).....	52
6.10 Outgoing-Call-Reply (OCRP).....	53
6.11 Outgoing-Call-Connected (OCCN).....	53
6.12 Call-Disconnect-Notify (CDN).....	53
6.13 WAN-Error-Notify (WEN).....	54
6.14 Set-Link-Info (SLI).....	54
7.0 Control Connection State Machines.....	54
7.1 Control Connection Protocol Operation.....	55
7.2 Control Connection States.....	56
7.2.1 Control Connection Establishment.....	56
7.3 Timing considerations.....	58
7.4 Incoming calls.....	58
7.4.1 LAC Incoming Call States.....	60
7.4.2 LNS Incoming Call States.....	62
7.5 Outgoing calls.....	63
7.5.1 LAC Outgoing Call States.....	64
7.5.2 LNS Outgoing Call States.....	66
7.6 Tunnel Disconnection.....	67
8.0 L2TP Over Specific Media.....	67
8.1 L2TP over UDP/IP.....	68

8.2 IP.....	69
9.0 Security Considerations.....	69
9.1 Tunnel Endpoint Security.....	70
9.2 Packet Level Security.....	70
9.3 End to End Security.....	70
9.4 L2TP and IPsec.....	71
9.5 Proxy PPP Authentication.....	71
10.0 IANA Considerations.....	71
10.1 AVP Attributes.....	71
10.2 Message Type AVP Values.....	72
10.3 Result Code AVP Values.....	72
10.3.1 Result Code Field Values.....	72
10.3.2 Error Code Field Values.....	72
10.4 Framing Capabilities & Bearer Capabilities.....	72
10.5 Proxy Authen Type AVP Values.....	72
10.6 AVP Header Bits.....	73
11.0 References.....	73
12.0 Acknowledgments.....	74
13.0 Authors' Addresses.....	75
Appendix A: Control Channel Slow Start and Congestion Avoidance.....	76
Appendix B: Control Message Examples.....	77
Appendix C: Intellectual Property Notice.....	79
Full Copyright Statement.....	80

## 1.0 Introduction

PPP [RFC1661] defines an encapsulation mechanism for transporting multiprotocol packets across layer 2 (L2) point-to-point links. Typically, a user obtains a L2 connection to a Network Access Server (NAS) using one of a number of techniques (e.g., dialup POTS, ISDN, ADSL, etc.) and then runs PPP over that connection. In such a configuration, the L2 termination point and PPP session endpoint reside on the same physical device (i.e., the NAS).

L2TP extends the PPP model by allowing the L2 and PPP endpoints to reside on different devices interconnected by a packet-switched network. With L2TP, a user has an L2 connection to an access concentrator (e.g., modem bank, ADSL DSLAM, etc.), and the concentrator then tunnels individual PPP frames to the NAS. This allows the actual processing of PPP packets to be divorced from the termination of the L2 circuit.

One obvious benefit of such a separation is that instead of requiring the L2 connection terminate at the NAS (which may require a long-distance toll charge), the connection may terminate at a (local) circuit concentrator, which then extends the logical PPP session over

a shared infrastructure such as frame relay circuit or the Internet. From the user's perspective, there is no functional difference between having the L2 circuit terminate in a NAS directly or using L2TP.

L2TP may also solve the multilink hunt-group splitting problem. Multilink PPP [RFC1990] requires that all channels composing a multilink bundle be grouped at a single Network Access Server (NAS). Due to its ability to project a PPP session to a location other than the point at which it was physically received, L2TP can be used to make all channels terminate at a single NAS. This allows multilink operation even when the calls are spread across distinct physical NASs.

This document defines the necessary control protocol for on-demand creation of tunnels between two nodes and the accompanying encapsulation for multiplexing multiple, tunneled PPP sessions.

## 1.1 Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 1.2 Terminology

### Analog Channel

A circuit-switched communication path which is intended to carry 3.1 kHz audio in each direction.

### Attribute Value Pair (AVP)

The variable length concatenation of a unique Attribute (represented by an integer) and a Value containing the actual value identified by the attribute. Multiple AVPs make up Control Messages which are used in the establishment, maintenance, and teardown of tunnels.

### Call

A connection (or attempted connection) between a Remote System and LAC. For example, a telephone call through the PSTN. A Call (Incoming or Outgoing) which is successfully established between a Remote System and LAC results in a corresponding L2TP Session within a previously established Tunnel between the LAC and LNS. (See also: Session, Incoming Call, Outgoing Call).

### Called Number

An indication to the receiver of a call as to what telephone number the caller used to reach it.

### Calling Number

An indication to the receiver of a call as to the telephone number of the caller.

### CHAP

Challenge Handshake Authentication Protocol [RFC1994], a PPP cryptographic challenge/response authentication protocol in which the cleartext password is not passed over the line.

### Control Connection

A control connection operates in-band over a tunnel to control the establishment, release, and maintenance of sessions and of the tunnel itself.

### Control Messages

Control messages are exchanged between LAC and LNS pairs, operating in-band within the tunnel protocol. Control messages govern aspects of the tunnel and sessions within the tunnel.

### Digital Channel

A circuit-switched communication path which is intended to carry digital information in each direction.

### DSLAM

Digital Subscriber Line (DSL) Access Module. A network device used in the deployment of DSL service. This is typically a concentrator of individual DSL lines located in a central office (CO) or local exchange.

### Incoming Call

A Call received at an LAC to be tunneled to an LNS (see Call, Outgoing Call).

### L2TP Access Concentrator (LAC)

A node that acts as one side of an L2TP tunnel endpoint and is a peer to the L2TP Network Server (LNS). The LAC sits between an LNS and a remote system and forwards packets to and from each. Packets sent from the LAC to the LNS requires tunneling with the L2TP protocol as defined in this document. The connection from the LAC to the remote system is either local (see: Client LAC) or a PPP link.

### L2TP Network Server (LNS)

A node that acts as one side of an L2TP tunnel endpoint and is a peer to the L2TP Access Concentrator (LAC). The LNS is the logical termination point of a PPP session that is being tunneled from the remote system by the LAC.

### Management Domain (MD)

A network or networks under the control of a single administration, policy or system. For example, an LNS's Management Domain might be the corporate network it serves. An LAC's Management Domain might be the Internet Service Provider that owns and manages it.

### Network Access Server (NAS)

A device providing local network access to users across a remote access network such as the PSTN. An NAS may also serve as an LAC, LNS or both.

### Outgoing Call

A Call placed by an LAC on behalf of an LNS (see Call, Incoming Call).

### Peer

When used in context with L2TP, peer refers to either the LAC or LNS. An LAC's Peer is an LNS and vice versa. When used in context with PPP, a peer is either side of the PPP connection.

### POTS

Plain Old Telephone Service.

### Remote System

An end-system or router attached to a remote access network (i.e. a PSTN), which is either the initiator or recipient of a call. Also referred to as a dial-up or virtual dial-up client.

### Session

L2TP is connection-oriented. The LNS and LAC maintain state for each Call that is initiated or answered by an LAC. An L2TP Session is created between the LAC and LNS when an end-to-end PPP connection is established between a Remote System and the LNS. Datagrams related to the PPP connection are sent over the Tunnel between the LAC and LNS. There is a one to one relationship between established L2TP Sessions and their associated Calls. (See also: Call).

### Tunnel

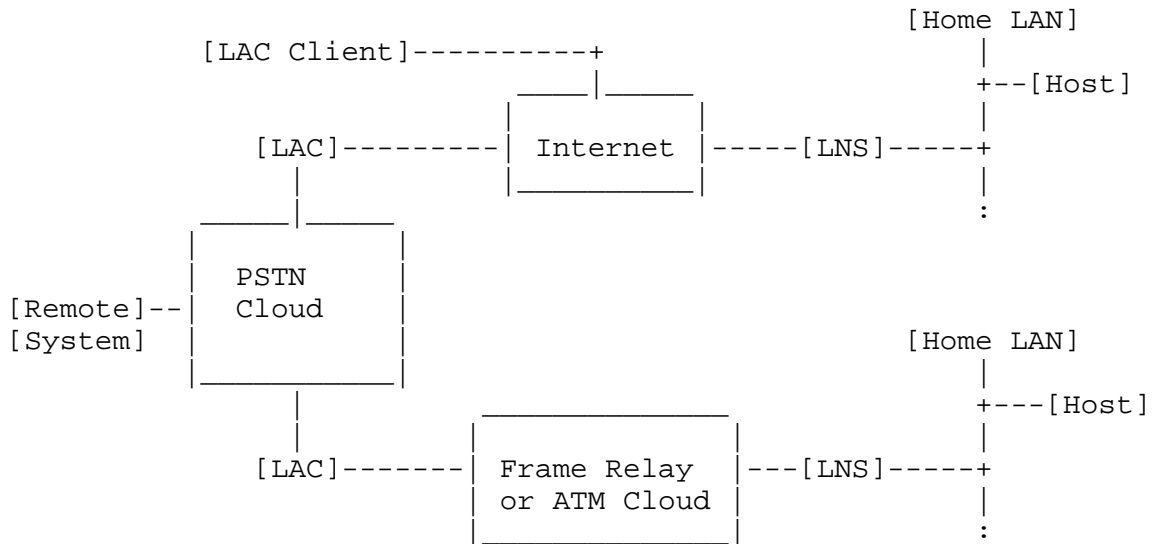
A Tunnel exists between a LAC-LNS pair. The Tunnel consists of a Control Connection and zero or more L2TP Sessions. The Tunnel carries encapsulated PPP datagrams and Control Messages between the LAC and the LNS.

### Zero-Length Body (ZLB) Message

A control packet with only an L2TP header. ZLB messages are used for explicitly acknowledging packets on the reliable control channel.

## 2.0 Topology

The following diagram depicts a typical L2TP scenario. The goal is to tunnel PPP frames between the Remote System or LAC Client and an LNS located at a Home LAN.



The Remote System initiates a PPP connection across the PSTN Cloud to an LAC. The LAC then tunnels the PPP connection across the Internet, Frame Relay, or ATM Cloud to an LNS whereby access to a Home LAN is obtained. The Remote System is provided addresses from the HOME LAN

via PPP NCP negotiation. Authentication, Authorization and Accounting may be provided by the Home LAN's Management Domain as if the user were connected to a Network Access Server directly.

A LAC Client (a Host which runs L2TP natively) may also participate in tunneling to the Home LAN without use of a separate LAC. In this case, the Host containing the LAC Client software already has a connection to the public Internet. A "virtual" PPP connection is then created and the local L2TP LAC Client software creates a tunnel to the LNS. As in the above case, Addressing, Authentication, Authorization and Accounting will be provided by the Home LAN's Management Domain.



### 3.0 Protocol Overview

L2TP utilizes two types of messages, control messages and data messages. Control messages are used in the establishment, maintenance and clearing of tunnels and calls. Data messages are used to encapsulate PPP frames being carried over the tunnel. Control messages utilize a reliable Control Channel within L2TP to guarantee delivery (see section 5.1 for details). Data messages are not retransmitted when packet loss occurs.

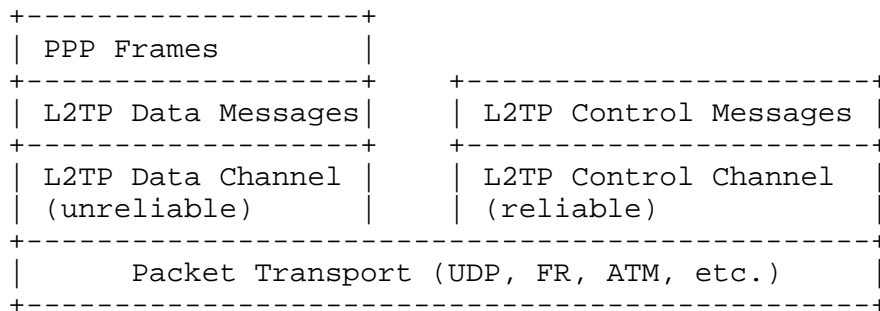


Figure 3.0 L2TP Protocol Structure

Figure 3.0 depicts the relationship of PPP frames and Control Messages over the L2TP Control and Data Channels. PPP Frames are passed over an unreliable Data Channel encapsulated first by an L2TP header and then a Packet Transport such as UDP, Frame Relay, ATM, etc. Control messages are sent over a reliable L2TP Control Channel which transmits packets in-band over the same Packet Transport.

Sequence numbers are required to be present in all control messages and are used to provide reliable delivery on the Control Channel. Data Messages may use sequence numbers to reorder packets and detect lost packets.

All values are placed into their respective fields and sent in network order (high order octets first).

#### 3.1 L2TP Header Format

L2TP packets for the control channel and data channel share a common header format. In each case where a field is optional, its space does not exist in the message if the field is marked not present. Note that while optional on data messages, the Length, Ns, and Nr fields marked as optional below, are required to be present on all control messages.

This header is formatted:

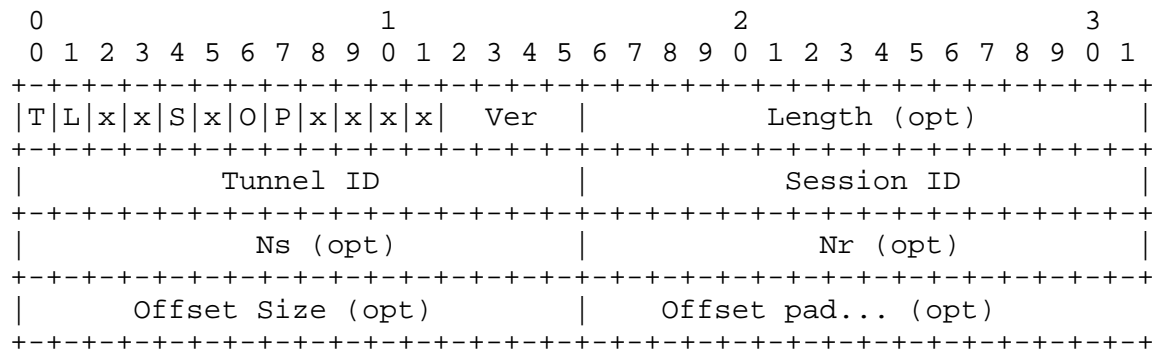


Figure 3.1 L2TP Message Header

The Type (T) bit indicates the type of message. It is set to 0 for a data message and 1 for a control message.

If the Length (L) bit is 1, the Length field is present. This bit MUST be set to 1 for control messages.

The x bits are reserved for future extensions. All reserved bits MUST be set to 0 on outgoing messages and ignored on incoming messages.

If the Sequence (S) bit is set to 1 the Ns and Nr fields are present. The S bit MUST be set to 1 for control messages.

If the Offset (O) bit is 1, the Offset Size field is present. The O bit MUST be set to 0 (zero) for control messages.

If the Priority (P) bit is 1, this data message should receive preferential treatment in its local queuing and transmission. LCP echo requests used as a keepalive for the link, for instance, should generally be sent with this bit set to 1. Without it, a temporary interval of local congestion could result in interference with keepalive messages and unnecessary loss of the link. This feature is only for use with data messages. The P bit MUST be set to 0 for all control messages.

Ver MUST be 2, indicating the version of the L2TP data message header described in this document. The value 1 is reserved to permit detection of L2F [RFC2341] packets should they arrive intermixed with L2TP packets. Packets received with an unknown Ver field MUST be discarded.

The Length field indicates the total length of the message in octets.

Tunnel ID indicates the identifier for the control connection. L2TP tunnels are named by identifiers that have local significance only. That is, the same tunnel will be given different Tunnel IDs by each end of the tunnel. Tunnel ID in each message is that of the intended recipient, not the sender. Tunnel IDs are selected and exchanged as Assigned Tunnel ID AVPs during the creation of a tunnel.

Session ID indicates the identifier for a session within a tunnel. L2TP sessions are named by identifiers that have local significance only. That is, the same session will be given different Session IDs by each end of the session. Session ID in each message is that of the intended recipient, not the sender. Session IDs are selected and exchanged as Assigned Session ID AVPs during the creation of a session.

Ns indicates the sequence number for this data or control message, beginning at zero and incrementing by one (modulo  $2^{16}$ ) for each message sent. See Section 5.8 and 5.4 for more information on using this field.

Nr indicates the sequence number expected in the next control message to be received. Thus, Nr is set to the Ns of the last in-order message received plus one (modulo  $2^{16}$ ). In data messages, Nr is reserved and, if present (as indicated by the S-bit), MUST be ignored upon receipt. See section 5.8 for more information on using this field in control messages.

The Offset Size field, if present, specifies the number of octets past the L2TP header at which the payload data is expected to start. Actual data within the offset padding is undefined. If the offset field is present, the L2TP header ends after the last octet of the offset padding.

### 3.2 Control Message Types

The Message Type AVP (see section 4.4.1) defines the specific type of control message being sent. Recall from section 3.1 that this is only for control messages, that is, messages with the T-bit set to 1.

This document defines the following control message types (see Section 6.1 through 6.14 for details on the construction and use of each message):

#### Control Connection Management

- 0 (reserved)
- 1 (SCCRQ) Start-Control-Connection-Request
- 2 (SCCRP) Start-Control-Connection-Reply
- 3 (SCCCN) Start-Control-Connection-Connected
- 4 (StopCCN) Stop-Control-Connection-Notification
- 5 (reserved)
- 6 (HELLO) Hello

#### Call Management

- 7 (OCRQ) Outgoing-Call-Request
- 8 (OCRP) Outgoing-Call-Reply
- 9 (OCCN) Outgoing-Call-Connected
- 10 (ICRQ) Incoming-Call-Request
- 11 (ICRP) Incoming-Call-Reply
- 12 (ICCN) Incoming-Call-Connected
- 13 (reserved)
- 14 (CDN) Call-Disconnect-Notify

#### Error Reporting

- 15 (WEN) WAN-Error-Notify

#### PPP Session Control

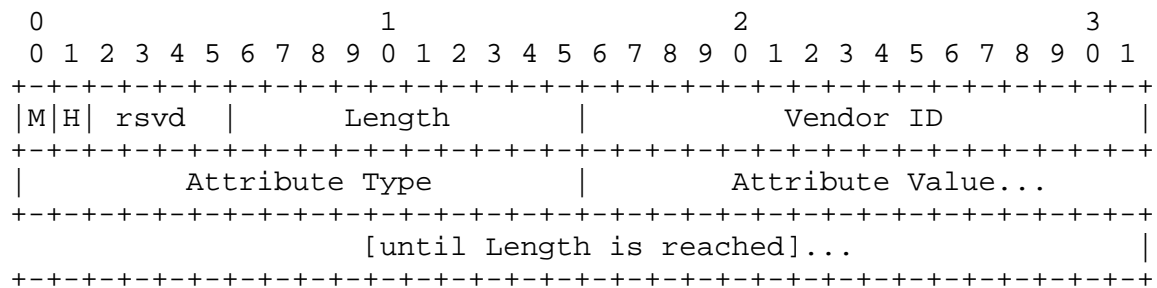
- 16 (SLI) Set-Link-Info

### 4.0 Control Message Attribute Value Pairs

To maximize extensibility while still permitting interoperability, a uniform method for encoding message types and bodies is used throughout L2TP. This encoding will be termed AVP (Attribute-Value Pair) in the remainder of this document.

## 4.1 AVP Format

Each AVP is encoded as:



The first six bits are a bit mask, describing the general attributes of the AVP.

Two bits are defined in this document, the remaining are reserved for future extensions. Reserved bits MUST be set to 0. An AVP received with a reserved bit set to 1 MUST be treated as an unrecognized AVP.

**Mandatory (M) bit:** Controls the behavior required of an implementation which receives an AVP which it does not recognize. If the M bit is set on an unrecognized AVP within a message associated with a particular session, the session associated with this message MUST be terminated. If the M bit is set on an unrecognized AVP within a message associated with the overall tunnel, the entire tunnel (and all sessions within) MUST be terminated. If the M bit is not set, an unrecognized AVP MUST be ignored. The control message must then continue to be processed as if the AVP had not been present.

**Hidden (H) bit:** Identifies the hiding of data in the Attribute Value field of an AVP. This capability can be used to avoid the passing of sensitive data, such as user passwords, as cleartext in an AVP. Section 4.3 describes the procedure for performing AVP hiding.

**Length:** Encodes the number of octets (including the Overall Length and bitmask fields) contained in this AVP. The Length may be calculated as 6 + the length of the Attribute Value field in octets. The field itself is 10 bits, permitting a maximum of 1023 octets of data in a single AVP. The minimum Length of an AVP is 6. If the length is 6, then the Attribute Value field is absent.

**Vendor ID:** The IANA assigned "SMI Network Management Private Enterprise Codes" [RFC1700] value. The value 0, corresponding to IETF adopted attribute values, is used for all AVPs defined within this document. Any vendor wishing to implement their own L2TP extensions can use their own Vendor ID along with private Attribute

values, guaranteeing that they will not collide with any other vendor's extensions, nor with future IETF extensions. Note that there are 16 bits allocated for the Vendor ID, thus limiting this feature to the first 65,535 enterprises.

**Attribute Type:** A 2 octet value with a unique interpretation across all AVPs defined under a given Vendor ID.

**Attribute Value:** This is the actual value as indicated by the Vendor ID and Attribute Type. It follows immediately after the Attribute Type field, and runs for the remaining octets indicated in the Length (i.e., Length minus 6 octets of header). This field is absent if the Length is 6.

#### 4.2 Mandatory AVPs

Receipt of an unknown AVP that has the M-bit set is catastrophic to the session or tunnel it is associated with. Thus, the M bit should only be defined for AVPs which are absolutely crucial to proper operation of the session or tunnel. Further, in the case where the LAC or LNS receives an unknown AVP with the M-bit set and shuts down the session or tunnel accordingly, it is the full responsibility of the peer sending the Mandatory AVP to accept fault for causing a non-interoperable situation. Before defining an AVP with the M-bit set, particularly a vendor-specific AVP, be sure that this is the intended consequence.

When an adequate alternative exists to use of the M-bit, it should be utilized. For example, rather than simply sending an AVP with the M-bit set to determine if a specific extension exists, availability may be identified by sending an AVP in a request message and expecting a corresponding AVP in a reply message.

Use of the M-bit with new AVPs (those not defined in this document) MUST provide the ability to configure the associated feature off, such that the AVP is either not sent, or sent with the M-bit not set.

#### 4.3 Hiding of AVP Attribute Values

The H bit in the header of each AVP provides a mechanism to indicate to the receiving peer whether the contents of the AVP are hidden or present in cleartext. This feature can be used to hide sensitive control message data such as user passwords or user IDs.

The H bit MUST only be set if a shared secret exists between the LAC and LNS. The shared secret is the same secret that is used for tunnel authentication (see Section 5.1.1). If the H bit is set in any

AVP(s) in a given control message, a Random Vector AVP must also be present in the message and MUST precede the first AVP having an H bit of 1.

Hiding an AVP value is done in several steps. The first step is to take the length and value fields of the original (cleartext) AVP and encode them into a Hidden AVP Subformat 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
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Length of Original Value | Original Attribute Value ...
+-----+-----+-----+-----+-----+-----+-----+-----+
| ... | Padding ...
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Length of Original Attribute Value: This is length of the Original Attribute Value to be obscured in octets. This is necessary to determine the original length of the Attribute Value which is lost when the additional Padding is added.

Original Attribute Value: Attribute Value that is to be obscured.

Padding: Random additional octets used to obscure length of the Attribute Value that is being hidden.

To mask the size of the data being hidden, the resulting subformat MAY be padded as shown above. Padding does NOT alter the value placed in the Length of Original Attribute Value field, but does alter the length of the resultant AVP that is being created. For example, If an Attribute Value to be hidden is 4 octets in length, the unhidden AVP length would be 10 octets (6 + Attribute Value length). After hiding, the length of the AVP will become 6 + Attribute Value length + size of the Length of Original Attribute Value field + Padding. Thus, if Padding is 12 octets, the AVP length will be 6 + 4 + 2 + 12 = 24 octets.

Next, An MD5 hash is performed on the concatenation of:

- + the 2 octet Attribute number of the AVP
- + the shared secret
- + an arbitrary length random vector

The value of the random vector used in this hash is passed in the value field of a Random Vector AVP. This Random Vector AVP must be placed in the message by the sender before any hidden AVPs. The same random vector may be used for more than one hidden AVP in the same

message. If a different random vector is used for the hiding of subsequent AVPs then a new Random Vector AVP must be placed in the command message before the first AVP to which it applies.

The MD5 hash value is then XORed with the first 16 octet (or less) segment of the Hidden AVP Subformat and placed in the Attribute Value field of the Hidden AVP. If the Hidden AVP Subformat is less than 16 octets, the Subformat is transformed as if the Attribute Value field had been padded to 16 octets before the XOR, but only the actual octets present in the Subformat are modified, and the length of the AVP is not altered.

If the Subformat is longer than 16 octets, a second one-way MD5 hash is calculated over a stream of octets consisting of the shared secret followed by the result of the first XOR. That hash is XORed with the second 16 octet (or less) segment of the Subformat and placed in the corresponding octets of the Value field of the Hidden AVP.

If necessary, this operation is repeated, with the shared secret used along with each XOR result to generate the next hash to XOR the next segment of the value with.

The hiding method was adapted from RFC 2138 [RFC2138] which was taken from the "Mixing in the Plaintext" section in the book "Network Security" by Kaufman, Perlman and Speciner [KPS]. A detailed explanation of the method follows:

Call the shared secret S, the Random Vector RV, and the Attribute Value AV. Break the value field into 16-octet chunks p1, p2, etc. with the last one padded at the end with random data to a 16-octet boundary. Call the ciphertext blocks c(1), c(2), etc. We will also define intermediate values b1, b2, etc.

$$\begin{array}{ll} b1 = \text{MD5}(\text{AV} + \text{S} + \text{RV}) & c(1) = p1 \text{ xor } b1 \\ b2 = \text{MD5}(\text{S} + c(1)) & c(2) = p2 \text{ xor } b2 \\ & \vdots \\ & \vdots \\ b_i = \text{MD5}(\text{S} + c(i-1)) & c(i) = p_i \text{ xor } b_i \end{array}$$

The String will contain c(1)+c(2)+...+c(i) where + denotes concatenation.

On receipt, the random vector is taken from the last Random Vector AVP encountered in the message prior to the AVP to be unhidden. The above process is then reversed to yield the original value.



#### 4.4 AVP Summary

The following sections contain a list of all L2TP AVPs defined in this document.

Following the name of the AVP is a list indicating the message types that utilize each AVP. After each AVP title follows a short description of the purpose of the AVP, a detail (including a graphic) of the format for the Attribute Value, and any additional information needed for proper use of the avp.

##### 4.4.1 AVPs Applicable To All Control Messages

###### Message Type (All Messages)

The Message Type AVP, Attribute Type 0, identifies the control message herein and defines the context in which the exact meaning of the following AVPs will be determined.

The Attribute Value field for this AVP has the following format:

```

      0                               1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+
|           Message Type           |
+---+---+---+---+---+---+---+---+---+

```

The Message Type is a 2 octet unsigned integer.

The Message Type AVP MUST be the first AVP in a message, immediately following the control message header (defined in section 3.1). See Section 3.2 for the list of defined control message types and their identifiers.

The Mandatory (M) bit within the Message Type AVP has special meaning. Rather than an indication as to whether the AVP itself should be ignored if not recognized, it is an indication as to whether the control message itself should be ignored. Thus, if the M-bit is set within the Message Type AVP and the Message Type is unknown to the implementation, the tunnel MUST be cleared. If the M-bit is not set, then the implementation may ignore an unknown message type. The M-bit MUST be set to 1 for all message types defined in this document. This AVP may not be hidden (the H-bit MUST be 0). The Length of this AVP is 8.

## Random Vector (All Messages)

The Random Vector AVP, Attribute Type 36, is used to enable the hiding of the Attribute Value of arbitrary AVPs.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Random Octet String ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Random Octet String may be of arbitrary length, although a random vector of at least 16 octets is recommended. The string contains the random vector for use in computing the MD5 hash to retrieve or hide the Attribute Value of a hidden AVP (see Section 4.2).

More than one Random Vector AVP may appear in a message, in which case a hidden AVP uses the Random Vector AVP most closely preceding it. This AVP MUST precede the first AVP with the H bit set.

The M-bit for this AVP MUST be set to 1. This AVP MUST NOT be hidden (the H-bit MUST be 0). The Length of this AVP is 6 plus the length of the Random Octet String.

## 4.4.2 Result and Error Codes

## Result Code (CDN, StopCCN)

The Result Code AVP, Attribute Type 1, indicates the reason for terminating the control channel or session.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Result Code           |           Error Code (opt)           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Error Message (opt) ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Result Code is a 2 octet unsigned integer. The optional Error Code is a 2 octet unsigned integer. An optional Error Message can follow the Error Code field. Presence of the Error Code and

Message are indicated by the AVP Length field. The Error Message contains an arbitrary string providing further (human readable) text associated with the condition. Human readable text in all error messages MUST be provided in the UTF-8 charset using the Default Language [RFC2277].

This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for this AVP MUST be set to 1. The Length is 8 if there is no Error Code or Message, 10 if there is an Error Code and no Error Message or 10 + the length of the Error Message if there is an Error Code and Message.

Defined Result Code values for the StopCCN message are:

- 0 - Reserved
- 1 - General request to clear control connection
- 2 - General error--Error Code indicates the problem
- 3 - Control channel already exists
- 4 - Requester is not authorized to establish a control channel
- 5 - The protocol version of the requester is not supported  
Error Code indicates highest version supported
- 6 - Requester is being shut down
- 7 - Finite State Machine error

Defined Result Code values for the CDN message are:

- 0 - Reserved
- 1 - Call disconnected due to loss of carrier
- 2 - Call disconnected for the reason indicated in error code
- 3 - Call disconnected for administrative reasons
- 4 - Call failed due to lack of appropriate facilities being available (temporary condition)
- 5 - Call failed due to lack of appropriate facilities being available (permanent condition)
- 6 - Invalid destination
- 7 - Call failed due to no carrier detected
- 8 - Call failed due to detection of a busy signal
- 9 - Call failed due to lack of a dial tone
- 10 - Call was not established within time allotted by LAC
- 11 - Call was connected but no appropriate framing was detected

The Error Codes defined below pertain to types of errors that are not specific to any particular L2TP request, but rather to protocol or message format errors. If an L2TP reply indicates in

its Result Code that a general error occurred, the General Error value should be examined to determine what the error was. The currently defined General Error codes and their meanings are:

- 0 - No general error
- 1 - No control connection exists yet for this LAC-LNS pair
- 2 - Length is wrong
- 3 - One of the field values was out of range or reserved field was non-zero
- 4 - Insufficient resources to handle this operation now
- 5 - The Session ID is invalid in this context
- 6 - A generic vendor-specific error occurred in the LAC
- 7 - Try another. If LAC is aware of other possible LNS destinations, it should try one of them. This can be used to guide an LAC based on LNS policy, for instance, the existence of multilink PPP bundles.
- 8 - Session or tunnel was shutdown due to receipt of an unknown AVP with the M-bit set (see section 4.2). The Error Message SHOULD contain the attribute of the offending AVP in (human readable) text form.

When a General Error Code of 6 is used, additional information about the error SHOULD be included in the Error Message field.

#### 4.4.3 Control Connection Management AVPs

##### Protocol Version (SCCRP, SCCRQ)

The Protocol Version AVP, Attribute Type 2, indicates the L2TP protocol version of the sender.

The Attribute Value field for this AVP has the following format:

```

      0                               1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Ver           |           Rev           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Ver field is a 1 octet unsigned integer containing the value 1. Rev field is a 1 octet unsigned integer containing 0. This pertains to L2TP protocol version 1, revision 0. Note this is not the same version number that is included in the header of each message.

This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for this AVP MUST be set to 1. The Length of this AVP is 8.

## Framing Capabilities (SCCRP, SCCRQ)

The Framing Capabilities AVP, Attribute Type 3, provides the peer with an indication of the types of framing that will be accepted or requested by the sender.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Reserved for future framing type definitions           |A|S|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Attribute Value field is a 32-bit mask, with two bits defined. If bit A is set, asynchronous framing is supported. If bit S is set, synchronous framing is supported.

A peer MUST NOT request an incoming or outgoing call with a Framing Type AVP specifying a value not advertised in the Framing Capabilities AVP it received during control connection establishment. Attempts to do so will result in the call being rejected.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) is 10.

## Bearer Capabilities (SCCRP, SCCRQ)

The Bearer Capabilities AVP, Attribute Type 4, provides the peer with an indication of the bearer device types supported by the hardware interfaces of the sender for outgoing calls.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Reserved for future bearer type definitions           |A|D|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

This is a 32-bit mask, with two bits defined. If bit A is set, analog access is supported. If bit D is set, digital access is supported.

An LNS should not request an outgoing call specifying a value in the Bearer Type AVP for a device type not advertised in the Bearer Capabilities AVP it received from the LAC during control connection establishment. Attempts to do so will result in the call being rejected.

This AVP MUST be present if the sender can place outgoing calls when requested.

Note that an LNS that cannot act as an LAC as well will not support hardware devices for handling incoming and outgoing calls and should therefore set the A and D bits of this AVP to 0, or should not send the AVP at all. An LNS that can also act as an LAC and place outgoing calls should set A or D as appropriate. Presence of this message is not a guarantee that a given outgoing call will be placed by the sender if requested, just that the physical capability exists.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) is 10.

#### Tie Breaker (SCCRQ)

The Tie Breaker AVP, Attribute Type 5, indicates that the sender wishes a single tunnel to exist between the given LAC-LNS pair.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Tie Break Value...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                                                                ...(64 bits)
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Tie Breaker Value is an 8 octet value that is used to choose a single tunnel where both LAC and LNS request a tunnel concurrently. The recipient of a SCCRQ must check to see if a SCCRQ has been sent to the peer, and if so, must compare its Tie Breaker value with the received one. The lower value "wins", and the "loser" MUST silently discard its tunnel. In the case where a tie breaker is present on both sides, and the value is equal, both sides MUST discard their tunnels.

If a tie breaker is received, and an outstanding SCCRP had no tie breaker value, the initiator which included the Tie Breaker AVP "wins". If neither side issues a tie breaker, then two separate tunnels are opened.

This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for this AVP MUST be set to 0. The Length of this AVP is 14.

#### Firmware Revision (SCCRP, SCCRPQ)

The Firmware Revision AVP, Attribute Type 6, indicates the firmware revision of the issuing device.

The Attribute Value field for this AVP has the following format:

```

      0                               1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+
|           Firmware Revision           |
+---+---+---+---+---+---+---+---+---+

```

The Firmware Revision is a 2 octet unsigned integer encoded in a vendor specific format.

For devices which do not have a firmware revision (general purpose computers running L2TP software modules, for instance), the revision of the L2TP software module may be reported instead.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) is 8.

#### Host Name (SCCRP, SCCRPQ)

The Host Name AVP, Attribute Type 7, indicates the name of the issuing LAC or LNS.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Host Name ... (arbitrary number of octets)
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Host Name is of arbitrary length, but MUST be at least 1 octet.

This name should be as broadly unique as possible; for hosts participating in DNS [RFC1034], a hostname with fully qualified domain would be appropriate.

This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for this AVP MUST be set to 1. The Length of this AVP is 6 plus the length of the Host Name.

#### Vendor Name (SCCRP, SCCRQ)

The Vendor Name AVP, Attribute Type 8, contains a vendor specific (possibly human readable) string describing the type of LAC or LNS being used.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Vendor Name ...(arbitrary number of octets)
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Vendor Name is the indicated number of octets representing the vendor string. Human readable text for this AVP MUST be provided in the UTF-8 charset using the Default Language [RFC2277].

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 6 plus the length of the Vendor Name.

#### Assigned Tunnel ID (SCCRP, SCCRQ, StopCCN)

The Assigned Tunnel ID AVP, Attribute Type 9, encodes the ID being assigned to this tunnel by the sender.

The Attribute Value field for this AVP has the following format:

```

      0                               1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Assigned Tunnel ID
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Assigned Tunnel ID is a 2 octet non-zero unsigned integer.

The Assigned Tunnel ID AVP establishes a value used to multiplex and demultiplex multiple tunnels between the LNS and LAC. The L2TP peer MUST place this value in the Tunnel ID header field of all



control and data messages that it subsequently transmits over the associated tunnel. Before the Assigned Tunnel ID AVP is received from a peer, messages MUST be sent to that peer with a Tunnel ID value of 0 in the header of all control messages.

In the StopCCN control message, the Assigned Tunnel ID AVP MUST be the same as the Assigned Tunnel ID AVP first sent to the receiving peer, permitting the peer to identify the appropriate tunnel even if a StopCCN is sent before an Assigned Tunnel ID AVP is received.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 8.

#### Receive Window Size (SCCRQ, SCCRP)

The Receive Window Size AVP, Attribute Type 10, specifies the receive window size being offered to the remote peer.

The Attribute Value field for this AVP has the following format:

```

      0                               1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+
|           Window Size           |
+---+---+---+---+---+---+---+---+

```

The Window Size is a 2 octet unsigned integer.

If absent, the peer must assume a Window Size of 4 for its transmit window. The remote peer may send the specified number of control messages before it must wait for an acknowledgment.

This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for this AVP MUST be set to 1. The Length of this AVP is 8.

#### Challenge (SCCRP, SCCRQ)

The Challenge AVP, Attribute Type 11, indicates that the issuing peer wishes to authenticate the tunnel endpoints using a CHAP-style authentication mechanism.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Challenge ... (arbitrary number of octets)
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Challenge is one or more octets of random data.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 6 plus the length of the Challenge.

#### Challenge Response (SCCCN, SCCRCP)

The Response AVP, Attribute Type 13, provides a response to a challenge received.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Response ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
... (16 octets)
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Response is a 16 octet value reflecting the CHAP-style [RFC1994] response to the challenge.

This AVP MUST be present in an SCCRCP or SCCCEN if a challenge was received in the preceding SCCRQ or SCCRCP. For purposes of the ID value in the CHAP response calculation, the value of the Message Type AVP for this message is used (e.g. 2 for an SCCRCP, and 3 for an SCCCEN).

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 22.

#### 4.4.4 Call Management AVPs

##### Q.931 Cause Code (CDN)

The Q.931 Cause Code AVP, Attribute Type 12, is used to give additional information in case of unsolicited call disconnection.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Cause Code           | Cause Msg   | Advisory Msg...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Cause Code is the returned Q.931 Cause code, and Cause Msg is the returned Q.931 message code (e.g., DISCONNECT) associated with the Cause Code. Both values are returned in their native ITU encodings [DSS1]. An additional ASCII text Advisory Message may also be included (presence indicated by the AVP Length) to further explain the reason for disconnecting.

This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for this AVP MUST be set to 1. The Length of this AVP is 9, plus the size of the Advisory Message.

##### Assigned Session ID (CDN, ICRP, ICRQ, OCRP, OCRQ)

The Assigned Session ID AVP, Attribute Type 14, encodes the ID being assigned to this session by the sender.

The Attribute Value field for this AVP has the following format:

```

      0                               1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Assigned Session ID   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Assigned Session ID is a 2 octet non-zero unsigned integer.

The Assigned Session ID AVP is establishes a value used to multiplex and demultiplex data sent over a tunnel between the LNS and LAC. The L2TP peer MUST place this value in the Session ID header field of all control and data messages that it subsequently transmits over the tunnel that belong to this session. Before the

Assigned Session ID AVP is received from a peer, messages MUST be sent to that peer with a Session ID of 0 in the header of all control messages.

In the CDN control message, the same Assigned Session ID AVP first sent to the receiving peer is used, permitting the peer to identify the appropriate tunnel even if CDN is sent before an Assigned Session ID is received.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 8.

#### Call Serial Number (ICRQ, OCRQ)

The Call Serial Number AVP, Attribute Type 15, encodes an identifier assigned by the LAC or LNS to this call.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Call Serial Number                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Call Serial Number is a 32 bit value.

The Call Serial Number is intended to be an easy reference for administrators on both ends of a tunnel to use when investigating call failure problems. Call Serial Numbers should be set to progressively increasing values, which are likely to be unique for a significant period of time across all interconnected LNSs and LACs.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 10.

#### Minimum BPS (OCRQ)

The Minimum BPS AVP, Attribute Type 16, encodes the lowest acceptable line speed for this call.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Minimum BPS                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Minimum BPS is a 32 bit value indicates the speed in bits per second.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 10.

#### Maximum BPS (OCRQ)

The Maximum BPS AVP, Attribute Type 17, encodes the highest acceptable line speed for this call.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Maximum BPS                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Maximum BPS is a 32 bit value indicates the speed in bits per second.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 10.

#### Bearer Type (ICRQ, OCRQ)

The Bearer Type AVP, Attribute Type 18, encodes the bearer type for the incoming or outgoing call.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               Reserved for future Bearer Types               |A|D|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Bearer Type is a 32-bit bit mask, which indicates the bearer capability of the call (ICRQ) or required for the call (OCRQ). If set, bit A indicates that the call refers to an analog channel. If set, bit D indicates that the call refers to a digital channel. Both may be set, indicating that the call was either indistinguishable, or can be placed on either type of channel.

Bits in the Value field of this AVP MUST only be set by the LNS for an OCRQ if it was set in the Bearer Capabilities AVP received from the LAC during control connection establishment.

It is valid to set neither the A nor D bits in an ICRQ. Such a setting may indicate that the call was not received over a physical link (e.g if the LAC and PPP are located in the same subsystem).

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 10.

#### Framing Type (ICCN, OCCN, OCRQ)

The Framing Type AVP, Attribute Type 19, encodes the framing type for the incoming or outgoing call.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Reserved for future Framing Types                               |A|S|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Framing Type is a 32-bit mask, which indicates the type of PPP framing requested for an OCRQ, or the type of PPP framing negotiated for an OCCN or ICCN. The framing type MAY be used as an indication to PPP on the LNS as to what link options to use for LCP negotiation [RFC1662].

Bit A indicates asynchronous framing. Bit S indicates synchronous framing. For an OCRQ, both may be set, indicating that either type of framing may be used.

Bits in the Value field of this AVP MUST only be set by the LNS for an OCRQ if it was set in the Framing Capabilities AVP received from the LAC during control connection establishment.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 10.

#### Called Number (ICRQ, OCRQ)

The Called Number AVP, Attribute Type 21, encodes the telephone number to be called for an OCRQ, and the Called number for an ICRQ.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Called Number... (arbitrary number of octets)                       |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Called Number is an ASCII string. Contact between the administrator of the LAC and the LNS may be necessary to coordinate interpretation of the value needed in this AVP.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 6 plus the length of the Called Number.

#### Calling Number (ICRQ)

The Calling Number AVP, Attribute Type 22, encodes the originating number for the incoming call.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Calling Number... (arbitrary number of octets)                       |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Calling Number is an ASCII string. Contact between the administrator of the LAC and the LNS may be necessary to coordinate interpretation of the value in this AVP.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 6 plus the length of the Calling Number.

## Sub-Address (ICRQ, OCRQ)

The Sub-Address AVP, Attribute Type 23, encodes additional dialing information.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Sub-Address ... (arbitrary number of octets)                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Sub-Address is an ASCII string. Contact between the administrator of the LAC and the LNS may be necessary to coordinate interpretation of the value in this AVP.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 6 plus the length of the Sub-Address.

## (Tx) Connect Speed (ICCN, OCCN)

The (Tx) Connect Speed BPS AVP, Attribute Type 24, encodes the speed of the facility chosen for the connection attempt.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               BPS                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The (Tx) Connect Speed BPS is a 4 octet value indicating the speed in bits per second.

When the optional Rx Connect Speed AVP is present, the value in this AVP represents the transmit connect speed, from the perspective of the LAC (e.g. data flowing from the LAC to the remote system). When the optional Rx Connect Speed AVP is NOT present, the connection speed between the remote system and LAC is assumed to be symmetric and is represented by the single value in this AVP.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 10.



## Rx Connect Speed (ICCN, OCCN)

The Rx Connect Speed AVP, Attribute Type 38, represents the speed of the connection from the perspective of the LAC (e.g. data flowing from the remote system to the LAC).

The Attribute Value field for this AVP has the following format:

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								
BPS (H)										BPS (L)																													

BPS is a 4 octet value indicating the speed in bits per second.

Presence of this AVP implies that the connection speed may be asymmetric with respect to the transmit connect speed given in the (Tx) Connect Speed AVP.

This AVP may be hidden (the H-bit MAY be 1 or 0). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 10.

## Physical Channel ID (ICRQ, OCRP)

The Physical Channel ID AVP, Attribute Type 25, encodes the vendor specific physical channel number used for a call.

The Attribute Value field for this AVP has the following format:

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								
Physical Channel ID																																							

Physical Channel ID is a 4 octet value intended to be used for logging purposes only.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 10.

### Private Group ID (ICCN)

The Private Group ID AVP, Attribute Type 37, is used by the LAC to indicate that this call is to be associated with a particular customer group.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Private Group ID ... (arbitrary number of octets)   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Private Group ID is a string of octets of arbitrary length.

The LNS MAY treat the PPP session as well as network traffic through this session in a special manner determined by the peer. For example, if the LNS is individually connected to several private networks using unregistered addresses, this AVP may be included by the LAC to indicate that a given call should be associated with one of the private networks.

The Private Group ID is a string corresponding to a table in the LNS that defines the particular characteristics of the selected group. A LAC MAY determine the Private Group ID from a RADIUS response, local configuration, or some other source.

This AVP may be hidden (the H-bit MAY be 1 or 0). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 6 plus the length of the Private Group ID.

### Sequencing Required (ICCN, OCCN)

The Sequencing Required AVP, Attribute Type 39, indicates to the LNS that Sequence Numbers MUST always be present on the data channel.

This AVP has no Attribute Value field.

This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for this AVP MUST be set to 1. The Length of this AVP is 6.

#### 4.4.5 Proxy LCP and Authentication AVPs

The LAC may have answered the call and negotiated LCP with the remote system, perhaps in order to establish the system's apparent identity. In this case, these AVPs may be included to indicate the

link properties the remote system initially requested, properties the remote system and LAC ultimately negotiated, as well as PPP authentication information sent and received by the LAC. This information may be used to initiate the PPP LCP and authentication systems on the LNS, allowing PPP to continue without renegotiation of LCP. Note that the LNS policy may be to enter an additional round of LCP negotiation and/or authentication if the LAC is not trusted.

#### Initial Received LCP CONFREQ (ICCN)

In the Initial Received LCP CONFREQ AVP, Attribute Type 26, provides the LNS with the Initial CONFREQ received by the LAC from the PPP Peer.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| LCP CONFREQ... (arbitrary number of octets)                        |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

LCP CONFREQ is a copy of the body of the initial CONFREQ received, starting at the first option within the body of the LCP message.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 6 plus the length of the CONFREQ.

#### Last Sent LCP CONFREQ (ICCN)

In the Last Sent LCP CONFREQ AVP, Attribute Type 27, provides the LNS with the Last CONFREQ sent by the LAC to the PPP Peer.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| LCP CONFREQ... (arbitrary number of octets)                        |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The LCP CONFREQ is a copy of the body of the final CONFREQ sent to the client to complete LCP negotiation, starting at the first option within the body of the LCP message.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 6 plus the length of the CONFREQ.

#### Last Received LCP CONFREQ (ICCN)

The Last Received LCP CONFREQ AVP, Attribute Type 28, provides the LNS with the Last CONFREQ received by the LAC from the PPP Peer.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| LCP CONFREQ... (arbitrary number of octets)                        |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The LCP CONFREQ is a copy of the body of the final CONFREQ received from the client to complete LCP negotiation, starting at the first option within the body of the LCP message.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 6 plus the length of the CONFREQ.

#### Proxy Authen Type (ICCN)

The Proxy Authen Type AVP, Attribute Type 29, determines if proxy authentication should be used.

The Attribute Value field for this AVP has the following format:

```

      0                               1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Authen Type                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Authen Type is a 2 octet unsigned integer, holding:

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 8.

Defined Authen Type values are:

- 0 - Reserved
- 1 - Textual username/password exchange
- 2 - PPP CHAP
- 3 - PPP PAP
- 4 - No Authentication
- 5 - Microsoft CHAP Version 1 (MSCHAPv1)

This AVP MUST be present if proxy authentication is to be utilized. If it is not present, then it is assumed that this peer cannot perform proxy authentication, requiring a restart of the authentication phase at the LNS if the client has already entered this phase with the LAC (which may be determined by the Proxy LCP AVP if present).

Associated AVPs for each type of authentication follow.

#### Proxy Authen Name (ICCN)

The Proxy Authen Name AVP, Attribute Type 30, specifies the name of the authenticating client when using proxy authentication.

The Attribute Value field for this AVP has the following format:

```

      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
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| Authen Name... (arbitrary number of octets)                        |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

Authen Name is a string of octets of arbitrary length. It contains the name specified in the client's authentication response.

This AVP MUST be present in messages containing a Proxy Authen Type AVP with an Authen Type of 1, 2, 3 or 5. It may be desirable to employ AVP hiding for obscuring the cleartext name.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) is 6 plus the length of the cleartext name.

#### Proxy Authen Challenge (ICCN)

The Proxy Authen Challenge AVP, Attribute Type 31, specifies the challenge sent by the LAC to the PPP Peer, when using proxy authentication.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Challenge... (arbitrary number of octets)                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Challenge is a string of one or more octets.

This AVP MUST be present for Proxy Authen Types 2 and 5. The Challenge field contains the CHAP challenge presented to the client by the LAC.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 6, plus the length of the Challenge.

#### Proxy Authen ID (ICCN)

The Proxy Authen ID AVP, Attribute Type 32, specifies the ID value of the PPP Authentication that was started between the LAC and the PPP Peer, when proxy authentication is being used.

The Attribute Value field for this AVP has the following format:

```

      0                               1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Reserved   |         ID         |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

ID is a 2 octet unsigned integer, the most significant octet MUST be 0.

The Proxy Authen ID AVP MUST be present for Proxy authen types 2, 3 and 5. For 2 and 5, the ID field contains the byte ID value presented to the client by the LAC in its Challenge. For 3, it is the Identifier value of the Authenticate-Request.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0.

#### Proxy Authen Response (ICCN)

The Proxy Authen Response AVP, Attribute Type 33, specifies the PPP Authentication response received by the LAC from the PPP Peer, when proxy authentication is used.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Response... (arbitrary number of octets) |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Response is a string of octets.

This AVP MUST be present for Proxy authen types 1, 2, 3 and 5. The Response field contains the client's response to the challenge. For Proxy authen types 2 and 5, this field contains the response value received by the LAC. For types 1 or 3, it contains the clear text password received from the client by the LAC. In the case of cleartext passwords, AVP hiding is recommended.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 6 plus the length of the Response.

#### 4.4.6 Call Status AVPs

##### Call Errors (WEN)

The Call Errors AVP, Attribute Type 34, is used by the LAC to send error information to the LNS.

The Attribute Value field for this AVP has the following format:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Reserved           |           CRC Errors (H)           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           CRC Errors (L)           |           Framing Errors (H)           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Framing Errors (L)           |           Hardware Overruns (H)           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Hardware Overruns (L)           |           Buffer Overruns (H)           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Buffer Overruns (L)           |           Time-out Errors (H)           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Time-out Errors (L)           |           Alignment Errors (H)           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Alignment Errors (L)           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The following fields are defined:

Reserved - Not used, MUST be 0  
 CRC Errors - Number of PPP frames received with CRC errors since call was established  
 Framing Errors - Number of improperly framed PPP packets received  
 Hardware Overruns - Number of receive buffer over-runs since call was established  
 Buffer Overruns - Number of buffer over-runs detected since call was established  
 Time-out Errors - Number of time-outs since call was established  
 Alignment Errors - Number of alignment errors since call was established

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 32.

#### ACCM (SLI)

The ACCM AVP, Attribute Type 35, is used by the LNS to inform LAC of the ACCM negotiated with the PPP Peer by the LNS.

The Attribute Value field for this AVP has the following format:

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								
Reserved										Send ACCM (H)																													
Send ACCM (L)										Receive ACCM (H)																													
Receive ACCM (L)																																							

Send ACCM and Receive ACCM are each 4 octet values preceded by a 2 octet reserved quantity. The send ACCM value should be used by the LAC to process packets it sends on the connection. The receive ACCM value should be used by the LAC to process incoming packets on the connection. The default values used by the LAC for both these fields are 0xFFFFFFFF. The LAC should honor these fields unless it has specific configuration information to indicate that the requested mask must be modified to permit operation.

This AVP may be hidden (the H-bit MAY be 1 or 0). The M-bit for this AVP MUST be set to 1. The Length of this AVP is 16.



## 5.0 Protocol Operation

The necessary setup for tunneling a PPP session with L2TP consists of two steps, (1) establishing the Control Connection for a Tunnel, and (2) establishing a Session as triggered by an incoming or outgoing call request. The Tunnel and corresponding Control Connection MUST be established before an incoming or outgoing call is initiated. An L2TP Session MUST be established before L2TP can begin to tunnel PPP frames. Multiple Sessions may exist across a single Tunnel and multiple Tunnels may exist between the same LAC and LNS.

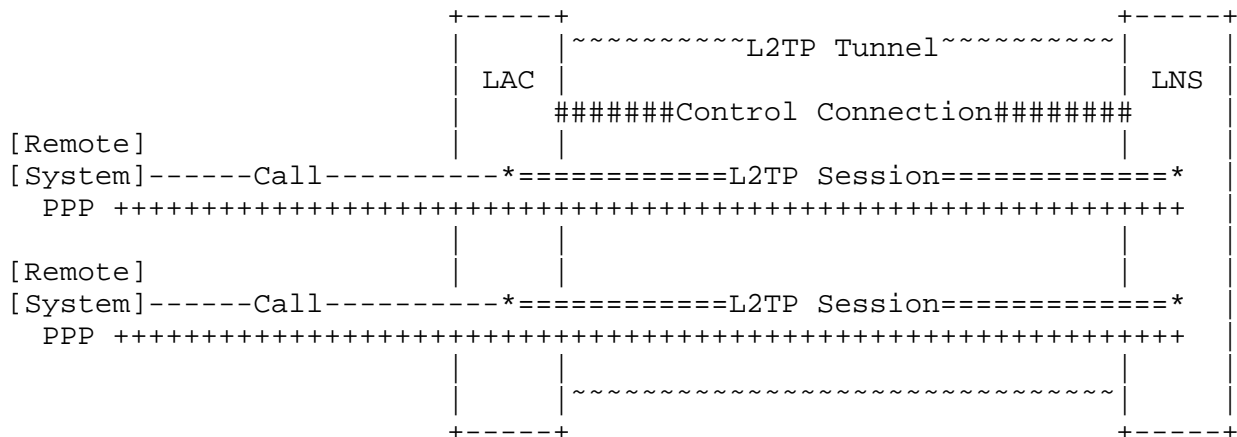


Figure 5.1 Tunneling PPP

### 5.1 Control Connection Establishment

The Control Connection is the initial connection that must be achieved between an LAC and LNS before sessions may be brought up. Establishment of the control connection includes securing the identity of the peer, as well as identifying the peer's L2TP version, framing, and bearer capabilities, etc.

A three message exchange is utilized to setup the control connection. Following is a typical message exchange:

```

LAC or LNS  LAC or LNS
-----
SCCRQ ->
          <- SCCRQ
SCCCN ->
          <- ZLB ACK

```

The ZLB ACK is sent if there are no further messages waiting in queue for that peer.

### 5.1.1 Tunnel Authentication

L2TP incorporates a simple, optional, CHAP-like [RFC1994] tunnel authentication system during control connection establishment. If an LAC or LNS wishes to authenticate the identity of the peer it is contacting or being contacted by, a Challenge AVP is included in the SCCRQ or SCCRP message. If a Challenge AVP is received in an SCCRQ or SCCRP, a Challenge Response AVP MUST be sent in the following SCCRP or SCCCEN, respectively. If the expected response and response received from a peer does not match, establishment of the tunnel MUST be disallowed.

To participate in tunnel authentication, a single shared secret MUST exist between the LAC and LNS. This is the same shared secret used for AVP hiding (see Section 4.3). See Section 4.4.3 for details on construction of the Challenge and Response AVPs.

## 5.2 Session Establishment

After successful control connection establishment, individual sessions may be created. Each session corresponds to single PPP stream between the LAC and LNS. Unlike control connection establishment, session establishment is directional with respect to the LAC and LNS. The LAC requests the LNS to accept a session for an incoming call, and the LNS requests the LAC to accept a session for placing an outgoing call.

### 5.2.1 Incoming Call Establishment

A three message exchange is employed to setup the session. Following is a typical sequence of events:

LAC	LNS
---	---
(Call Detected)	
ICRQ ->	
	<- ICRP
ICCN ->	
	<- ZLB ACK

The ZLB ACK is sent if there are no further messages waiting in queue for that peer.

### 5.2.2 Outgoing Call Establishment

A three message exchange is employed to setup the session. Following is a typical sequence of events:

```
LAC          LNS
---          ---
              <- OCRQ
OCRP ->

(Perform
 Call
 Operation)

OCCN ->
              <- ZLB ACK
```

The ZLB ACK is sent if there are no further messages waiting in queue for that peer.

### 5.3 Forwarding PPP Frames

Once tunnel establishment is complete, PPP frames from the remote system are received at the LAC, stripped of CRC, link framing, and transparency bytes, encapsulated in L2TP, and forwarded over the appropriate tunnel. The LNS receives the L2TP packet, and processes the encapsulated PPP frame as if it were received on a local PPP interface.

The sender of a message associated with a particular session and tunnel places the Session ID and Tunnel ID (specified by its peer) in the Session ID and Tunnel ID header for all outgoing messages. In this manner, PPP frames are multiplexed and demultiplexed over a single tunnel between a given LNS-LAC pair. Multiple tunnels may exist between a given LNS-LAC pair, and multiple sessions may exist within a tunnel.

The value of 0 for Session ID and Tunnel ID is special and MUST NOT be used as an Assigned Session ID or Assigned Tunnel ID. For the cases where a Session ID has not yet been assigned by the peer (i.e., during establishment of a new session or tunnel), the Session ID field MUST be sent as 0, and the Assigned Session ID AVP within the message MUST be used to identify the session. Similarly, for cases where the Tunnel ID has not yet been assigned from the peer, the Tunnel ID MUST be sent as 0 and Assigned Tunnel ID AVP used to identify the tunnel.

#### 5.4 Using Sequence Numbers on the Data Channel

Sequence numbers are defined in the L2TP header for control messages and optionally for data messages (see Section 3.1). These are used to provide a reliable control message transport (see Section 5.8) and optional data message sequencing. Each peer maintains separate sequence numbers for the control connection and each individual data session within a tunnel.

Unlike the L2TP control channel, the L2TP data channel does not use sequence numbers to retransmit lost data messages. Rather, data messages may use sequence numbers to detect lost packets and/or restore the original sequence of packets that may have been reordered during transport. The LAC may request that sequence numbers be present in data messages via the Sequencing Required AVP (see Section 4.4.6). If this AVP is present during session setup, sequence numbers **MUST** be present at all times. If this AVP is not present, sequencing presence is under control of the LNS. The LNS controls enabling and disabling of sequence numbers by sending a data message with or without sequence numbers present at any time during the life of a session. Thus, if the LAC receives a data message without sequence numbers present, it **MUST** stop sending sequence numbers in future data messages. If the LAC receives a data message with sequence numbers present, it **MUST** begin sending sequence numbers in future outgoing data messages. If the LNS enables sequencing after disabling it earlier in the session, the sequence number state picks up where it left off before.

The LNS may initiate disabling of sequencing at any time during the session (including the first data message sent). It is recommended that for connections where reordering or packet loss may occur, sequence numbers always be enabled during the initial negotiation stages of PPP and disabled only when and if the risk is considered acceptable. For example, if the PPP session being tunneled is not utilizing any stateful compression or encryption protocols and is only carrying IP (as determined by the PPP NCPs that are established), then the LNS might decide to disable sequencing as IP is tolerant to datagram loss and reordering.

#### 5.5 Keepalive (Hello)

A keepalive mechanism is employed by L2TP in order to differentiate tunnel outages from extended periods of no control or data activity on a tunnel. This is accomplished by injecting Hello control messages (see Section 6.5) after a specified period of time has elapsed since the last data or control message was received on a tunnel. As for any other control message, if the Hello message is not reliably delivered then the tunnel is declared down and is reset. The transport reset

mechanism along with the injection of Hello messages ensures that a connectivity failure between the LNS and the LAC will be detected at both ends of a tunnel.

### 5.6 Session Teardown

Session teardown may be initiated by either the LAC or LNS and is accomplished by sending a CDN control message. After the last session is cleared, the control connection MAY be torn down as well (and typically is). Following is an example of a typical control message exchange:

```
LAC or LNS  LAC or LNS

CDN ->
(Clean up)

      <- ZLB ACK
          (Clean up)
```

### 5.7 Control Connection Teardown

Control connection teardown may be initiated by either the LAC or LNS and is accomplished by sending a single StopCCN control message. The receiver of a StopCCN MUST send a ZLB ACK to acknowledge receipt of the message and maintain enough control connection state to properly accept StopCCN retransmissions over at least a full retransmission cycle (in case the ZLB ACK is lost). The recommended time for a full retransmission cycle is 31 seconds (see section 5.8). Following is an example of a typical control message exchange:

```
LAC or LNS  LAC or LNS

StopCCN ->
(Clean up)

      <- ZLB ACK
          (Wait)
          (Clean up)
```

An implementation may shut down an entire tunnel and all sessions on the tunnel by sending the StopCCN. Thus, it is not necessary to clear each session individually when tearing down the whole tunnel.

## 5.8 Reliable Delivery of Control Messages

L2TP provides a lower level reliable transport service for all control messages. The Nr and Ns fields of the control message header (see section 3.1) belong to this transport. The upper level functions of L2TP are not concerned with retransmission or ordering of control messages. The reliable control message is a sliding window transport that provides control message retransmission and congestion control. Each peer maintains separate sequence number state for the control connection within a tunnel.

The message sequence number, Ns, begins at 0. Each subsequent message is sent with the next increment of the sequence number. The sequence number is thus a free running counter represented modulo 65536. The sequence number in the header of a received message is considered less than or equal to the last received number if its value lies in the range of the last received number and the preceding 32767 values, inclusive. For example, if the last received sequence number was 15, then messages with sequence numbers 0 through 15, as well as 32784 through 65535, would be considered less than or equal. Such a message would be considered a duplicate of a message already received and ignored from processing. However, in order to ensure that all messages are acknowledged properly (particularly in the case of a lost ZLB ACK message), receipt of duplicate messages MUST be acknowledged by the reliable transport. This acknowledgement may either piggybacked on a message in queue, or explicitly via a ZLB ACK.

All control messages take up one slot in the control message sequence number space, except the ZLB acknowledgement. Thus, Ns is not incremented after a ZLB message is sent.

The last received message number, Nr, is used to acknowledge messages received by an L2TP peer. It contains the sequence number of the message the peer expects to receive next (e.g. the last Ns of a non-ZLB message received plus 1, modulo 65536). While the Nr in a received ZLB is used to flush messages from the local retransmit queue (see below), Nr of the next message sent is not be updated by the Ns of the ZLB.

The reliable transport at a receiving peer is responsible for making sure that control messages are delivered in order and without duplication to the upper level. Messages arriving out of order may be queued for in-order delivery when the missing messages are received, or they may be discarded requiring a retransmission by the peer.

Each tunnel maintains a queue of control messages to be transmitted to its peer. The message at the front of the queue is sent with a given *Ns* value, and is held until a control message arrives from the peer in which the *Nr* field indicates receipt of this message. After a period of time (a recommended default is 1 second) passes without acknowledgement, the message is retransmitted. The retransmitted message contains the same *Ns* value, but the *Nr* value MUST be updated with the sequence number of the next expected message.

Each subsequent retransmission of a message MUST employ an exponential backoff interval. Thus, if the first retransmission occurred after 1 second, the next retransmission should occur after 2 seconds has elapsed, then 4 seconds, etc. An implementation MAY place a cap upon the maximum interval between retransmissions. This cap MUST be no less than 8 seconds per retransmission. If no peer response is detected after several retransmissions, (a recommended default is 5, but SHOULD be configurable), the tunnel and all sessions within MUST be cleared.

When a tunnel is being shut down for reasons other than loss of connectivity, the state and reliable delivery mechanisms MUST be maintained and operated for the full retransmission interval after the final message exchange has occurred.

A sliding window mechanism is used for control message transmission. Consider two peers A & B. Suppose A specifies a Receive Window Size AVP with a value of *N* in the SCCRQ or SCCRP messages. B is now allowed to have up to *N* outstanding control messages. Once *N* have been sent, it must wait for an acknowledgment that advances the window before sending new control messages. An implementation may support a receive window of only 1 (i.e., by sending out a Receive Window Size AVP with a value of 1), but MUST accept a window of up to 4 from its peer (e.g. have the ability to send 4 messages before backing off). A value of 0 for the Receive Window Size AVP is invalid.

When retransmitting control messages, a slow start and congestion avoidance window adjustment procedure SHOULD be utilized. The recommended procedure for this is described in Appendix A.

A peer MUST NOT withhold acknowledgment of messages as a technique for flow controlling control messages. An L2TP implementation is expected to be able to keep up with incoming control messages, possibly responding to some with errors reflecting an inability to honor the requested action.

Appendix B contains examples of control message transmission, acknowledgement, and retransmission.

## 6.0 Control Connection Protocol Specification

The following control connection messages are used to establish, clear and maintain L2TP tunnels. All data is sent in network order (high order octets first). Any "reserved" or "empty" fields MUST be sent as 0 values to allow for protocol extensibility.

### 6.1 Start-Control-Connection-Request (SCCRQ)

Start-Control-Connection-Request (SCCRQ) is a control message used to initialize a tunnel between an LNS and an LAC. It is sent by either the LAC or the LNS to begin the tunnel establishment process.

The following AVPs MUST be present in the SCCRQ:

- Message Type AVP
- Protocol Version
- Host Name
- Framing Capabilities
- Assigned Tunnel ID

The Following AVPs MAY be present in the SCCRQ:

- Bearer Capabilities
- Receive Window Size
- Challenge
- Tie Breaker
- Firmware Revision
- Vendor Name

### 6.2 Start-Control-Connection-Reply (SCCRP)

Start-Control-Connection-Reply (SCCRP) is a control message sent in reply to a received SCCRQ message. SCCRP is used to indicate that the SCCRQ was accepted and establishment of the tunnel should continue.

The following AVPs MUST be present in the SCCRP:

- Message Type
- Protocol Version
- Framing Capabilities
- Host Name
- Assigned Tunnel ID



The following AVPs MAY be present in the SCCRPs:

- Bearer Capabilities
- Firmware Revision
- Vendor Name
- Receive Window Size
- Challenge
- Challenge Response

### 6.3 Start-Control-Connection-Connected (SCCCN)

Start-Control-Connection-Connected (SCCCN) is a control message sent in reply to an SCCRPs. SCCCn completes the tunnel establishment process.

The following AVP MUST be present in the SCCCn:

- Message Type

The following AVP MAY be present in the SCCCn:

- Challenge Response

### 6.4 Stop-Control-Connection-Notification (StopCCN)

Stop-Control-Connection-Notification (StopCCN) is a control message sent by either the LAC or LNS to inform its peer that the tunnel is being shutdown and the control connection should be closed. In addition, all active sessions are implicitly cleared (without sending any explicit call control messages). The reason for issuing this request is indicated in the Result Code AVP. There is no explicit reply to the message, only the implicit ACK that is received by the reliable control message transport layer.

The following AVPs MUST be present in the StopCCN:

- Message Type
- Assigned Tunnel ID
- Result Code

### 6.5 Hello (HELLO)

The Hello (HELLO) message is an L2TP control message sent by either peer of a LAC-LNS control connection. This control message is used as a "keepalive" for the tunnel.

The sending of HELLO messages and the policy for sending them are left up to the implementation. A peer **MUST NOT** expect HELLO messages at any time or interval. As with all messages sent on the control connection, the receiver will return either a ZLB ACK or an (unrelated) message piggybacking the necessary acknowledgement information.

Since a HELLO is a control message, and control messages are reliably sent by the lower level transport, this keepalive function operates by causing the transport level to reliably deliver a message. If a media interruption has occurred, the reliable transport will be unable to deliver the HELLO across, and will clean up the tunnel.

Keepalives for the tunnel **MAY** be implemented by sending a HELLO if a period of time (a recommended default is 60 seconds, but **SHOULD** be configurable) has passed without receiving any message (data or control) from the peer.

HELLO messages are global to the tunnel. The Session ID in a HELLO message **MUST** be 0.

The Following AVP **MUST** be present in the HELLO message:

Message Type

#### 6.6 Incoming-Call-Request (ICRQ)

Incoming-Call-Request (ICRQ) is a control message sent by the LAC to the LNS when an incoming call is detected. It is the first in a three message exchange used for establishing a session within an L2TP tunnel.

ICRQ is used to indicate that a session is to be established between the LAC and LNS for this call and provides the LNS with parameter information for the session. The LAC may defer answering the call until it has received an ICRP from the LNS indicating that the session should be established. This mechanism allows the LNS to obtain sufficient information about the call before determining whether it should be answered or not. Alternatively, the LAC may answer the call, negotiate LCP and PPP authentication, and use the information gained to choose the LNS. In this case, the call has already been answered by the time the ICRP message is received; the LAC simply spoofs the "call indication" and "call answer" steps in this case.

The following AVPs MUST be present in the ICRQ:

- Message Type
- Assigned Session ID
- Call Serial Number

The following AVPs MAY be present in the ICRQ:

- Bearer Type
- Physical Channel ID
- Calling Number
- Called Number
- Sub-Address

#### 6.7 Incoming-Call-Reply (ICRP)

Incoming-Call-Reply (ICRP) is a control message sent by the LNS to the LAC in response to a received ICRQ message. It is the second in the three message exchange used for establishing sessions within an L2TP tunnel.

ICRP is used to indicate that the ICRQ was successful and for the LAC to answer the call if it has not already done so. It also allows the LNS to indicate necessary parameters for the L2TP session.

The following AVPs MUST be present in the ICRP:

- Message Type
- Assigned Session ID

#### 6.8 Incoming-Call-Connected (ICCN)

Incoming-Call-Connected (ICCN) is a control message sent by the LAC to the LNS in response to a received ICRP message. It is the third message in the three message exchange used for establishing sessions within an L2TP tunnel.

ICCN is used to indicate that the ICRP was accepted, the call has been answered, and that the L2TP session should move to the established state. It also provides additional information to the LNS about parameters used for the answered call (parameters that may not always be available at the time the ICRQ is issued).

The following AVPs MUST be present in the ICCN:

- Message Type
- (Tx) Connect Speed
- Framing Type

The following AVPs MAY be present in the ICCN:

- Initial Received LCP CONFREQ
- Last Sent LCP CONFREQ
- Last Received LCP CONFREQ
- Proxy Authen Type
- Proxy Authen Name
- Proxy Authen Challenge
- Proxy Authen ID
- Proxy Authen Response
- Private Group ID
- Rx Connect Speed
- Sequencing Required

#### 6.9 Outgoing-Call-Request (OCRQ)

Outgoing-Call-Request (OCRQ) is a control message sent by the LNS to the LAC to indicate that an outbound call from the LAC is to be established. It is the first in a three message exchange used for establishing a session within an L2TP tunnel.

OCRQ is used to indicate that a session is to be established between the LNS and LAC for this call and provides the LAC with parameter information for both the L2TP session, and the call that is to be placed

An LNS MUST have received a Bearer Capabilities AVP during tunnel establishment from an LAC in order to request an outgoing call to that LAC.

The following AVPs MUST be present in the OCRQ:

- Message Type
- Assigned Session ID
- Call Serial Number
- Minimum BPS
- Maximum BPS
- Bearer Type
- Framing Type
- Called Number

The following AVPs MAY be present in the OCRQ:

- Sub-Address

### 6.10 Outgoing-Call-Reply (OCRP)

Outgoing-Call-Reply (OCRP) is a control message sent by the LAC to the LNS in response to a received OCRQ message. It is the second in a three message exchange used for establishing a session within an L2TP tunnel.

OCRP is used to indicate that the LAC is able to attempt the outbound call and returns certain parameters regarding the call attempt.

The following AVPs MUST be present in the OCRP:

- Message Type
- Assigned Session ID

The following AVPs MAY be present in the OCRP:

- Physical Channel ID

### 6.11 Outgoing-Call-Connected (OCCN)

Outgoing-Call-Connected (OCCN) is a control message sent by the LAC to the LNS following the OCRP and after the outgoing call has been completed. It is the final message in a three message exchange used for establishing a session within an L2TP tunnel.

OCCN is used to indicate that the result of a requested outgoing call was successful. It also provides information to the LNS about the particular parameters obtained after the call was established.

The following AVPs MUST be present in the OCCN:

- Message Type
- (Tx) Connect Speed
- Framing Type

The following AVPs MAY be present in the OCCN:

- Rx Connect Speed
- Sequencing Required

### 6.12 Call-Disconnect-Notify (CDN)

The Call-Disconnect-Notify (CDN) message is an L2TP control message sent by either the LAC or LNS to request disconnection of a specific call within the tunnel. Its purpose is to inform the peer of the

disconnection and the reason why the disconnection occurred. The peer MUST clean up any resources, and does not send back any indication of success or failure for such cleanup.

The following AVPs MUST be present in the CDN:

- Message Type
- Result Code
- Assigned Session ID

The following AVPs MAY be present in the CDN:

- Q.931 Cause Code

#### 6.13 WAN-Error-Notify (WEN)

The WAN-Error-Notify message is an L2TP control message sent by the LAC to the LNS to indicate WAN error conditions (conditions that occur on the interface supporting PPP). The counters in this message are cumulative. This message should only be sent when an error occurs, and not more than once every 60 seconds. The counters are reset when a new call is established.

The following AVPs MUST be present in the WEN:

- Message Type
- Call Errors

#### 6.14 Set-Link-Info (SLI)

The Set-Link-Info message is an L2TP control message sent by the LNS to the LAC to set PPP-negotiated options. These options can change at any time during the life of the call, thus the LAC MUST be able to update its internal call information and behavior on an active PPP session.

The following AVPs MUST be present in the SLI:

- Message Type
- ACCM

### 7.0 Control Connection State Machines

The control messages defined in section 6 are exchanged by way of state tables defined in this section. Tables are defined for incoming call placement, outgoing call placement, as well as for initiation of

the tunnel itself. The state tables do not encode timeout and retransmission behavior, as this is handled in the underlying semantics defined in Section 5.8.

## 7.1 Control Connection Protocol Operation

This section describes the operation of various L2TP control connection functions and the Control Connection messages which are used to support them.

Receipt of an invalid or unrecoverable malformed control message should be logged appropriately and the control connection cleared to ensure recovery to a known state. The control connection may then be restarted by the initiator.

An invalid control message is defined as a message which contains a Message Type that is marked mandatory (see Section 4.4.1) and is unknown to the implementation, or a control message that is received in an improper sequence (e.g. an SCCCN sent in reply to an SCCRQ).

Examples of a malformed control message include one that has an invalid value in its header, contains an AVP that is formatted incorrectly or whose value is out of range, or a message that is missing a required AVP. A control message with a malformed header should be discarded. A control message with an invalid AVP should look to the M-bit for that AVP to determine whether the error is recoverable or not.

A malformed yet recoverable non-mandatory (M-bit is not set) AVP within a control message should be treated in a similar manner as an unrecognized non-mandatory AVP. Thus, if a malformed AVP is received with the M-bit set, the session or tunnel should be terminated with a proper Result or Error Code sent. If the M-bit is not set, the AVP should be ignored (with the exception of logging a local error message) and the message accepted.

This MUST NOT be considered a license to send malformed AVPs, but simply a guide towards how to handle an improperly formatted message if one is received. It is impossible to list all potential malformations of a given message and give advice for each. That said, one example of a recoverable, malformed AVP might be if the Rx Connect Speed AVP, attribute 38, is received with a length of 8 rather than 10 and the BPS given in 2 octets rather than 4. Since the Rx Connect Speed is non-mandatory, this condition should not be considered catastrophic. As such, the control message should be accepted as if the AVP had not been received (with the exception of a local error message being logged).

In several cases in the following tables, a protocol message is sent, and then a "clean up" occurs. Note that regardless of the initiator of the tunnel destruction, the reliable delivery mechanism must be allowed to run (see Section 5.8) before destroying the tunnel. This permits the tunnel management messages to be reliably delivered to the peer.

Appendix B.1 contains an example of lock-step tunnel establishment.

## 7.2 Control Connection States

The L2TP control connection protocol is not distinguishable between the LNS and LAC, but is distinguishable between the originator and receiver. The originating peer is the one which first initiates establishment of the tunnel (in a tie breaker situation, this is the winner of the tie). Since either LAC or LNS can be the originator, a collision can occur. See the Tie Breaker AVP in Section 4.4.3 for a description of this and its resolution.

### 7.2.1 Control Connection Establishment

State -----	Event -----	Action -----	New State -----
idle	Local Open request	Send SCCRQ	wait-ctl-reply
idle	Receive SCCRQ, acceptable	Send SCCRP	wait-ctl-conn
idle	Receive SCCRQ, not acceptable	Send StopCCN, Clean up	idle
idle	Receive SCCRP	Send StopCCN Clean up	idle
idle	Receive SCCCN	Clean up	idle
wait-ctl-reply	Receive SCCRP, acceptable	Send SCCCN, Send tunnel-open event to waiting sessions	established
wait-ctl-reply	Receive SCCRP, not acceptable	Send StopCCN, Clean up	idle
wait-ctl-reply	Receive SCCRQ, lose tie-breaker	Clean up, Re-queue SCCRQ for idle state	idle



wait-ctl-reply	Receive SCCCN	Send StopCCN Clean up	idle
wait-ctl-conn	Receive SCCCN, acceptable	Send tunnel-open event to waiting sessions	established
wait-ctl-conn	Receive SCCCN, not acceptable	Send StopCCN, Clean up	idle
wait-ctl-conn	Receive SCCRP, SCCRQ	Send StopCCN, Clean up	idle
established	Local Open request (new call)	Send tunnel-open event to waiting sessions	established
established	Admin Tunnel Close	Send StopCCN Clean up	idle
established	Receive SCCRP, SCCRQ, SCCCN	Send StopCCN Clean up	idle
idle wait-ctl-reply, wait-ctl-conn, established	Receive StopCCN	Clean up	idle

The states associated with the LNS or LAC for control connection establishment are:

#### idle

Both initiator and recipient start from this state. An initiator transmits an SCCRP, while a recipient remains in the idle state until receiving an SCCRP.

#### wait-ctl-reply

The originator checks to see if another connection has been requested from the same peer, and if so, handles the collision situation described in Section 5.8.

When an SCCRP is received, it is examined for a compatible version. If the version of the reply is lower than the version sent in the request, the older (lower) version should be used provided it is supported. If the version in the reply is earlier and supported, the originator moves to the established state. If

the version is earlier and not supported, a StopCCN MUST be sent to the peer and the originator cleans up and terminates the tunnel.

#### wait-ctl-conn

This is where an SCCC is awaited; upon receipt, the challenge response is checked. The tunnel either is established, or is torn down if an authorization failure is detected.

#### established

An established connection may be terminated by either a local condition or the receipt of a Stop-Control-Connection-Notification. In the event of a local termination, the originator MUST send a Stop-Control-Connection-Notification and clean up the tunnel.

If the originator receives a Stop-Control-Connection-Notification it MUST also clean up the tunnel.

### 7.3 Timing considerations

Due to the real-time nature of telephone signaling, both the LNS and LAC should be implemented with multi-threaded architectures such that messages related to multiple calls are not serialized and blocked. The call and connection state figures do not specify exceptions caused by timers. These are addressed in Section 5.8.

### 7.4 Incoming calls

An Incoming-Call-Request message is generated by the LAC when an incoming call is detected (for example, an associated telephone line rings). The LAC selects a Session ID and serial number and indicates the call bearer type. Modems should always indicate analog call type. ISDN calls should indicate digital when unrestricted digital service or rate adaption is used and analog if digital modems are involved. Calling Number, Called Number, and Subaddress may be included in the message if they are available from the telephone network.

Once the LAC sends the Incoming-Call-Request, it waits for a response from the LNS but it does not necessarily answer the call from the telephone network yet. The LNS may choose not to accept the call if:

- No resources are available to handle more sessions
- The dialed, dialing, or subaddress fields do not correspond to an authorized user
- The bearer service is not authorized or supported

If the LNS chooses to accept the call, it responds with an Incoming-Call-Reply. When the LAC receives the Incoming-Call-Reply, it attempts to connect the call. A final call connected message from the LAC to the LNS indicates that the call states for both the LAC and the LNS should enter the established state. If the call terminated before the LNS could accept it, a Call-Disconnect-Notify is sent by the LAC to indicate this condition.

When the dialed-in client hangs up, the call is cleared normally and the LAC sends a Call-Disconnect-Notify message. If the LNS wishes to clear a call, it sends a Call-Disconnect-Notify message and cleans up its session.

## 7.4.1 LAC Incoming Call States

State -----	Event -----	Action -----	New State -----
idle	Bearer Ring or Ready to indicate incoming conn.	Initiate local tunnel open	wait-tunnel
idle	Receive ICCN, ICRP, CDN	Clean up	idle
wait-tunnel	Bearer line drop or local close request	Clean up	idle
wait-tunnel	tunnel-open	Send ICRQ	wait-reply
wait-reply	Receive ICRP, acceptable	Send ICCN	established
wait-reply	Receive ICRP, Not acceptable	Send CDN, Clean up	idle
wait-reply	Receive ICRQ	Send CDN Clean up	idle
wait-reply	Receive CDN ICCN	Clean up	idle
wait-reply	Local close request or Bearer line drop	Send CDN, Clean up	idle
established	Receive CDN	Clean up	idle
established	Receive ICRQ, ICRP, ICCN	Send CDN, Clean up	idle
established	Bearer line drop or local close request	Send CDN, Clean up	idle

The states associated with the LAC for incoming calls are:

idle

The LAC detects an incoming call on one of its interfaces. Typically this means an analog line is ringing or an ISDN TE has detected an incoming Q.931 SETUP message. The LAC initiates its tunnel establishment state machine, and moves to a state waiting for confirmation of the existence of a tunnel.

wait-tunnel

In this state the session is waiting for either the control connection to be opened or for verification that the tunnel is already open. Once an indication that the tunnel has/was opened, session control messages may be exchanged. The first of these is the Incoming-Call-Request.

wait-reply

The LAC receives either a CDN message indicating the LNS is not willing to accept the call (general error or don't accept) and moves back into the idle state, or an Incoming-Call-Reply message indicating the call is accepted, the LAC sends an Incoming-Call-Connected message and enters the established state.

established

Data is exchanged over the tunnel. The call may be cleared following:

- + An event on the connected interface: The LAC sends a Call-Disconnect-Notify message
- + Receipt of a Call-Disconnect-Notify message: The LAC cleans up, disconnecting the call.
- + A local reason: The LAC sends a Call-Disconnect-Notify message.

## 7.4.2 LNS Incoming Call States

State -----	Event -----	Action -----	New State -----
idle	Receive ICRQ, acceptable	Send ICRP	wait-connect
idle	Receive ICRQ, not acceptable	Send CDN, Clean up	idle
idle	Receive ICRP	Send CDN Clean up	idle
idle	Receive ICCN	Clean up	idle
wait-connect	Receive ICCN acceptable	Prepare for data	established
wait-connect	Receive ICCN not acceptable	Send CDN, Clean up	idle
wait-connect	Receive ICRQ, ICRP	Send CDN Clean up	idle
idle, wait-connect, established	Receive CDN	Clean up	idle
wait-connect established	Local Close request	Send CDN, Clean up	idle
established	Receive ICRQ, ICRP, ICCN	Send CDN Clean up	idle

The states associated with the LNS for incoming calls are:

## idle

An Incoming-Call-Request message is received. If the request is not acceptable, a Call-Disconnect-Notify is sent back to the LAC and the LNS remains in the idle state. If the Incoming-Call-Request message is acceptable, an Incoming-Call-Reply is sent. The session moves to the wait-connect state.

## wait-connect

If the session is still connected on the LAC, the LAC sends an Incoming-Call-Connected message to the LNS which then moves into established state. The LAC may send a Call-Disconnect-Notify to indicate that the incoming caller could not be connected. This

could happen, for example, if a telephone user accidentally places a standard voice call to an LAC resulting in a handshake failure on the called modem.

established

The session is terminated either by receipt of a Call-Disconnect-Notify message from the LAC or by sending a Call-Disconnect-Notify. Clean up follows on both sides regardless of the initiator.

## 7.5 Outgoing calls

Outgoing calls are initiated by an LNS and instruct an LAC to place a call. There are three messages for outgoing calls: Outgoing-Call-Request, Outgoing-Call-Reply, and Outgoing-Call-Connected. The LNS sends an Outgoing-Call-Request specifying the dialed party phone number, subaddress and other parameters. The LAC MUST respond to the Outgoing-Call-Request message with an Outgoing-Call-Reply message once the LAC determines that the proper facilities exist to place the call and the call is administratively authorized. For example, is this LNS allowed to dial an international call? Once the outbound call is connected, the LAC sends an Outgoing-Call-Connected message to the LNS indicating the final result of the call attempt:

## 7.5.1 LAC Outgoing Call States

State -----	Event -----	Action -----	New State -----
idle	Receive OCRQ, acceptable	Send OCRP, Open bearer	wait-cs-answer
idle	Receive OCRQ, not acceptable	Send CDN, Clean up	idle
idle	Receive OCRP	Send CDN Clean up	idle
idle	Receive OCCN, CDN	Clean up	idle
wait-cs-answer	Bearer answer, framing detected	Send OCCN	established
wait-cs-answer	Bearer failure	Send CDN, Clean up	idle
wait-cs-answer	Receive OCRQ, OCRP, OCCN	Send CDN Clean up	idle
established	Receive OCRQ, OCRP, OCCN	Send CDN Clean up	idle
wait-cs-answer, established	Receive CDN	Clean up	idle
established	Bearer line drop, Local close request	Send CDN, Clean up	idle

The states associated with the LAC for outgoing calls are:

## idle

If Outgoing-Call-Request is received in error, respond with a Call-Disconnect-Notify. Otherwise, allocate a physical channel and send an Outgoing-Call-Reply. Place the outbound call and move to the wait-cs-answer state.

## wait-cs-answer

If the call is not completed or a timer expires waiting for the call to complete, send a Call-Disconnect-Notify with the appropriate error condition set and go to idle state. If a circuit



switched connection is established and framing is detected, send an Outgoing-Call-Connected indicating success and go to established state.

established

If a Call-Disconnect-Notify is received by the LAC, the telco call MUST be released via appropriate mechanisms and the session cleaned up. If the call is disconnected by the client or the called interface, a Call-Disconnect-Notify message MUST be sent to the LNS. The sender of the Call-Disconnect-Notify message returns to the idle state after sending of the message is complete.

## 7.5.2 LNS Outgoing Call States

State -----	Event -----	Action -----	New State -----
idle	Local open request	Initiate local tunnel-open	wait-tunnel
idle	Receive OCCN, OCRP, CDN	Clean up	idle
wait-tunnel	tunnel-open	Send OCRQ	wait-reply
wait-reply	Receive OCRP, acceptable	none	wait-connect
wait-reply	Receive OCRP, not acceptable	Send CDN Clean up	idle
wait-reply	Receive OCCN, OCRQ	Send CDN Clean up	idle
wait-connect	Receive OCCN	none	established
wait-connect	Receive OCRQ, OCRP	Send CDN Clean up	idle
idle, wait-reply, wait-connect, established	Receive CDN,	Clean up	idle
established	Receive OCRQ, OCRP, OCCN	Send CDN Clean up	idle
wait-reply, wait-connect, established	Local Close request	Send CDN Clean up	idle
wait-tunnel	Local Close request	Clean up	idle

The states associated with the LNS for outgoing calls are:

idle, wait-tunnel

When an outgoing call is initiated, a tunnel is first created, much as the idle and wait-tunnel states for an LAC incoming call. Once a tunnel is established, an Outgoing-Call-Request message is sent to the LAC and the session moves into the wait-reply state.

#### wait-reply

If a Call-Disconnect-Notify is received, an error occurred, and the session is cleaned up and returns to idle. If an Outgoing-Call-Reply is received, the call is in progress and the session moves to the wait-connect state.

#### wait-connect

If a Call-Disconnect-Notify is received, the call failed; the session is cleaned up and returns to idle. If an Outgoing-Call-Connected is received, the call has succeeded and the session may now exchange data.

#### established

If a Call-Disconnect-Notify is received, the call has been terminated for the reason indicated in the Result and Cause Codes; the session moves back to the idle state. If the LNS chooses to terminate the session, it sends a Call-Disconnect-Notify to the LAC and then cleans up and idles its session.

### 7.6 Tunnel Disconnection

The disconnection of a tunnel consists of either peer issuing a Stop-Control-Connection-Notification. The sender of this Notification should wait a finite period of time for the acknowledgment of this message before releasing the control information associated with the tunnel. The recipient of this Notification should send an acknowledgment of the Notification and then release the associated control information.

When to release a tunnel is an implementation issue and is not specified in this document. A particular implementation may use whatever policy is appropriate for determining when to release a control connection. Some implementations may leave a tunnel open for a period of time or perhaps indefinitely after the last session for that tunnel is cleared. Others may choose to disconnect the tunnel immediately after the last user connection on the tunnel disconnects.

### 8.0 L2TP Over Specific Media

L2TP is self-describing, operating at a level above the media over which it is carried. However, some details of its connection to media are required to permit interoperable implementations. The following sections describe details needed to permit interoperability over specific media.

## 8.1 L2TP over UDP/IP

L2TP uses the registered UDP port 1701 [RFC1700]. The entire L2TP packet, including payload and L2TP header, is sent within a UDP datagram. The initiator of an L2TP tunnel picks an available source UDP port (which may or may not be 1701), and sends to the desired destination address at port 1701. The recipient picks a free port on its own system (which may or may not be 1701), and sends its reply to the initiator's UDP port and address, setting its own source port to the free port it found. Once the source and destination ports and addresses are established, they MUST remain static for the life of the tunnel.

It has been suggested that having the recipient choose an arbitrary source port (as opposed to using the destination port in the packet initiating the tunnel, i.e., 1701) may make it more difficult for L2TP to traverse some NAT devices. Implementors should consider the potential implication of this before choosing an arbitrary source port.

IP fragmentation may occur as the L2TP packet travels over the IP substrate. L2TP makes no special efforts to optimize this. A LAC implementation MAY cause its LCP to negotiate for a specific MRU, which could optimize for LAC environments in which the MTU's of the path over which the L2TP packets are likely to travel have a consistent value.

The default for any L2TP implementation is that UDP checksums MUST be enabled for both control and data messages. An L2TP implementation MAY provide an option to disable UDP checksums for data messages. It is recommended that UDP checksums always be enabled on control packets.

Port 1701 is used for both L2F [RFC2341] and L2TP packets. The Version field in each header may be used to discriminate between the two packet types (L2F uses a value of 1, and the L2TP version described in this document uses a value of 2). An L2TP implementation running on a system which does not support L2F MUST silently discard all L2F packets.

To the PPP clients using an L2TP-over-UDP/IP tunnel, the PPP link has the characteristic of being able to reorder or silently drop packets. The former may break non-IP protocols being carried by PPP, especially LAN-centric ones such as bridging. The latter may break protocols which assume per-packet indication of error, such as TCP header compression. Sequencing may be handled by using L2TP data message sequence numbers if any protocol being transported by the PPP

tunnel cannot tolerate reordering. The sequence dependency characteristics of individual protocols are outside the scope of this document.

Allowing packets to be dropped silently is perhaps more problematic with some protocols. If PPP reliable delivery [RFC1663] is enabled, no upper PPP protocol will encounter lost packets. If L2TP sequence numbers are enabled, L2TP can detect the packet loss. In the case of an LNS, the PPP and L2TP stacks are both present within the LNS, and packet loss signaling may occur precisely as if a packet was received with a CRC error. Where the LAC and PPP stack are co-resident, this technique also applies. Where the LAC and PPP client are physically distinct, the analogous signaling MAY be accomplished by sending a packet with a CRC error to the PPP client. Note that this would greatly increase the complexity of debugging client line problems, since the client statistics could not distinguish between true media errors and LAC-initiated ones. Further, this technique is not possible on all hardware.

If VJ compression is used, and neither PPP reliable delivery nor sequence numbers are enabled, each lost packet results in a 1 in  $2^{16}$  chance of a TCP segment being forwarded with incorrect contents [RFC1144]. Where the combination of the packet loss rate with this statistical exposure is unacceptable, TCP header compression SHOULD NOT be used.

In general, it is wise to remember that the L2TP/UDP/IP transport is an unreliable transport. As with any PPP media that is subject to loss, care should be taken when using protocols that are particularly loss-sensitive. Such protocols include compression and encryption protocols that employ history.

## 8.2 IP

When operating in IP environments, L2TP MUST offer the UDP encapsulation described in 8.1 as its default configuration for IP operation. Other configurations (perhaps corresponding to a compressed header format) MAY be defined and made available as a configurable option.

## 9.0 Security Considerations

L2TP encounters several security issues in its operation. The general approach of L2TP to these issues is documented here.

## 9.1 Tunnel Endpoint Security

The tunnel endpoints may optionally perform an authentication procedure of one another during tunnel establishment. This authentication has the same security attributes as CHAP, and has reasonable protection against replay and snooping during the tunnel establishment process. This mechanism is not designed to provide any authentication beyond tunnel establishment; it is fairly simple for a malicious user who can snoop the tunnel stream to inject packets once an authenticated tunnel establishment has been completed successfully.

For authentication to occur, the LAC and LNS MUST share a single secret. Each side uses this same secret when acting as authenticatee as well as authenticator. Since a single secret is used, the tunnel authentication AVPs include differentiating values in the CHAP ID fields for each message digest calculation to guard against replay attacks.

The Assigned Tunnel ID and Assigned Session ID (See Section 4.4.3) SHOULD be selected in an unpredictable manner rather than sequentially or otherwise. Doing so will help deter hijacking of a session by a malicious user who does not have access to packet traces between the LAC and LNS.

## 9.2 Packet Level Security

Securing L2TP requires that the underlying transport make available encryption, integrity and authentication services for all L2TP traffic. This secure transport operates on the entire L2TP packet and is functionally independent of PPP and the protocol being carried by PPP. As such, L2TP is only concerned with confidentiality, authenticity, and integrity of the L2TP packets between its tunnel

endpoints (the LAC and LNS), not unlike link-layer encryption being concerned only about protecting the confidentiality of traffic between its physical endpoints.

## 9.3 End to End Security

Protecting the L2TP packet stream via a secure transport does, in turn, also protect the data within the tunneled PPP packets while transported from the LAC to the LNS. Such protection should not be considered a substitution for end-to-end security between communicating hosts or applications.

#### 9.4 L2TP and IPsec

When running over IP, IPsec provides packet-level security via ESP and/or AH. All L2TP control and data packets for a particular tunnel appear as homogeneous UDP/IP data packets to the IPsec system.

In addition to IP transport security, IPsec defines a mode of operation that allows tunneling of IP packets. The packet level encryption and authentication provided by IPsec tunnel mode and that provided by L2TP secured with IPsec provide an equivalent level of security for these requirements.

IPsec also defines access control features that are required of a compliant IPsec implementation. These features allow filtering of packets based upon network and transport layer characteristics such as IP address, ports, etc. In the L2TP tunneling model, analogous filtering is logically performed at the PPP layer or network layer above L2TP. These network layer access control features may be handled at the LNS via vendor-specific authorization features based upon the authenticated PPP user, or at the network layer itself by using IPsec transport mode end-to-end between the communicating hosts. The requirements for access control mechanisms are not a part of the L2TP specification and as such are outside the scope of this document.

#### 9.5 Proxy PPP Authentication

L2TP defines AVPs that MAY be exchanged during session establishment to provide forwarding of PPP authentication information obtained at the LAC to the LNS for validation (see Section 4.4.5). This implies a direct trust relationship of the LAC on behalf of the LNS. If the LNS chooses to implement proxy authentication, it MUST be able to be configured off, requiring a new round a PPP authentication initiated by the LNS (which may or may not include a new round of LCP negotiation).

#### 10.0 IANA Considerations

This document defines a number of "magic" numbers to be maintained by the IANA. This section explains the criteria to be used by the IANA to assign additional numbers in each of these lists. The following subsections describe the assignment policy for the namespaces defined elsewhere in this document.

##### 10.1 AVP Attributes

As defined in Section 4.1, AVPs contain vendor ID, Attribute and Value fields. For vendor ID value of 0, IANA will maintain a registry

of assigned Attributes and in some case also values. Attributes 0-39 are assigned as defined in Section 4.4. The remaining values are available for assignment through IETF Consensus [RFC 2434].

## 10.2 Message Type AVP Values

As defined in Section 4.4.1, Message Type AVPs (Attribute Type 0) have an associated value maintained by IANA. Values 0-16 are defined in Section 3.2, the remaining values are available for assignment via IETF Consensus [RFC 2434]

## 10.3 Result Code AVP Values

As defined in Section 4.4.2, Result Code AVPs (Attribute Type 1) contain three fields. Two of these fields (the Result Code and Error Code fields) have associated values maintained by IANA.

### 10.3.1 Result Code Field Values

The Result Code AVP may be included in CDN and StopCCN messages. The allowable values for the Result Code field of the AVP differ depending upon the value of the Message Type AVP. For the StopCCN message, values 0-7 are defined in Section 4.4.2; for the StopCCN message, values 0-11 are defined in the same section. The remaining values of the Result Code field for both messages are available for assignment via IETF Consensus [RFC 2434].

### 10.3.2 Error Code Field Values

Values 0-7 are defined in Section 4.4.2. Values 8-32767 are available for assignment via IETF Consensus [RFC 2434]. The remaining values of the Error Code field are available for assignment via First Come First Served [RFC 2434].

## 10.4 Framing Capabilities & Bearer Capabilities

The Framing Capabilities AVP and Bearer Capabilities AVPs (defined in Section 4.4.3) both contain 32-bit bitmasks. Additional bits should only be defined via a Standards Action [RFC 2434].

## 10.5 Proxy Authen Type AVP Values

The Proxy Authen Type AVP (Attribute Type 29) has an associated value maintained by IANA. Values 0-5 are defined in Section 4.4.5, the remaining values are available for assignment via First Come First Served [RFC 2434].



## 10.6 AVP Header Bits

There are four remaining reserved bits in the AVP header. Additional bits should only be assigned via a Standards Action [RFC 2434].

## 11.0 References

- [DSS1] ITU-T Recommendation, "Digital subscriber Signaling System No. 1 (DSS 1) - ISDN user-network interface layer 3 specification for basic call control", Rec. Q.931(I.451), May 1998
- [KPS] Kaufman, C., Perlman, R., and Speciner, M., "Network Security: Private Communications in a Public World", Prentice Hall, March 1995, ISBN 0-13-061466-1
- [RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC1034] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987.
- [RFC1144] Jacobson, V., "Compressing TCP/IP Headers for Low-Speed Serial Links", RFC 1144, February 1990.
- [RFC1661] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.
- [RFC1662] Simpson, W., "PPP in HDLC-like Framing", STD 51, RFC 1662, July 1994.
- [RFC1663] Rand, D., "PPP Reliable Transmission", RFC 1663, July 1994.
- [RFC1700] Reynolds, J. and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994. See also:  
<http://www.iana.org/numbers.html>
- [RFC1990] Sklower, K., Lloyd, B., McGregor, G., Carr, D. and T. Coradetti, "The PPP Multilink Protocol (MP)", RFC 1990, August 1996.
- [RFC1994] Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996.
- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2138] Rigney, C., Rubens, A., Simpson, W. and S. Willens, "Remote Authentication Dial In User Service (RADIUS)", RFC 2138, April 1997.
- [RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.
- [RFC2341] Valencia, A., Littlewood, M. and T. Kolar, "Cisco Layer Two Forwarding (Protocol) L2F", RFC 2341, May 1998.
- [RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.
- [RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.
- [RFC2637] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W. and G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)", RFC 2637, July 1999.
- [STEVENS] Stevens, W. Richard, "TCP/IP Illustrated, Volume I The Protocols", Addison-Wesley Publishing Company, Inc., March 1996, ISBN 0-201-63346-9

## 12.0 Acknowledgments

The basic concept for L2TP and many of its protocol constructs were adopted from L2F [RFC2341] and PPTP [PPTP]. Authors of these are A. Valencia, M. Littlewood, T. Kolar, K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn.

Dory Leifer made valuable refinements to the protocol definition of L2TP and contributed to the editing of this document.

Steve Cobb and Evan Caves redesigned the state machine tables.

Barney Wolff provided a great deal of design input on the endpoint authentication mechanism.

John Bray, Greg Burns, Rich Garrett, Don Grosser, Matt Holdrege, Terry Johnson, Dory Leifer, and Rich Shea provided valuable input and review at the 43rd IETF in Orlando, FL., which led to improvement of the overall readability and clarity of this document.

### 13.0 Authors' Addresses

Gurdeep Singh Pall  
Microsoft Corporation  
Redmond, WA

EMail: gurdeep@microsoft.com

Bill Palter  
RedBack Networks, Inc  
1389 Moffett Park Drive  
Sunnyvale, CA 94089

EMail: palter@zev.net

Allan Rubens  
Ascend Communications  
1701 Harbor Bay Parkway  
Alameda, CA 94502

EMail: acr@del.com

W. Mark Townsley  
cisco Systems  
7025 Kit Creek Road  
PO Box 14987  
Research Triangle Park, NC 27709

EMail: townsley@cisco.com

Andrew J. Valencia  
cisco Systems  
170 West Tasman Drive  
San Jose CA 95134-1706

EMail: vandys@cisco.com

Glen Zorn  
Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052

EMail: gwz@acm.org

## Appendix A: Control Channel Slow Start and Congestion Avoidance

Although each side has indicated the maximum size of its receive window, it is recommended that a slow start and congestion avoidance method be used to transmit control packets. The methods described here are based upon the TCP congestion avoidance algorithm as described in section 21.6 of TCP/IP Illustrated, Volume I, by W. Richard Stevens [STEVENS].

Slow start and congestion avoidance make use of several variables. The congestion window (CWND) defines the number of packets a sender may send before waiting for an acknowledgment. The size of CWND expands and contracts as described below. Note however, that CWND is never allowed to exceed the size of the advertised window obtained from the Receive Window AVP (in the text below, it is assumed any increase will be limited by the Receive Window Size). The variable SSTHRESH determines when the sender switches from slow start to congestion avoidance. Slow start is used while CWND is less than SSHTRESH.

A sender starts out in the slow start phase. CWND is initialized to one packet, and SSHTRESH is initialized to the advertised window (obtained from the Receive Window AVP). The sender then transmits one packet and waits for its acknowledgement (either explicit or piggybacked). When the acknowledgement is received, the congestion window is incremented from one to two. During slow start, CWND is increased by one packet each time an ACK (explicit ZLB or piggybacked) is received. Increasing CWND by one on each ACK has the effect of doubling CWND with each round trip, resulting in an exponential increase. When the value of CWND reaches SSHTRESH, the slow start phase ends and the congestion avoidance phase begins.

During congestion avoidance, CWND expands more slowly. Specifically, it increases by  $1/\text{CWND}$  for every new ACK received. That is, CWND is increased by one packet after CWND new ACKs have been received. Window expansion during the congestion avoidance phase is effectively linear, with CWND increasing by one packet each round trip.

When congestion occurs (indicated by the triggering of a retransmission) one half of the CWND is saved in SSTHRESH, and CWND is set to one. The sender then reenters the slow start phase.

## Appendix B: Control Message Examples

## B.1: Lock-step tunnel establishment

In this example, an LAC establishes a tunnel, with the exchange involving each side alternating in sending messages. This example shows the final acknowledgment explicitly sent within a ZLB ACK message. An alternative would be to piggyback the acknowledgement within a message sent as a reply to the ICRQ or OCRQ that will likely follow from the side that initiated the tunnel.

LAC or LNS -----	LNS or LAC -----
SCCRQ       -> Nr: 0, Ns: 0	
	<-       SCCRP Nr: 1, Ns: 0
S CCCN       -> Nr: 1, Ns: 1	
	<-       ZLB Nr: 2, Ns: 1

## B.2: Lost packet with retransmission

An existing tunnel has a new session requested by the LAC. The ICRP is lost and must be retransmitted by the LNS. Note that loss of the ICRP has two impacts: not only does it keep the upper level state machine from progressing, but it also keeps the LAC from seeing a timely lower level acknowledgment of its ICRQ.

LAC ---	LNS ---
ICRQ       -> Nr: 1, Ns: 2	
	(packet lost)   <-       ICRP Nr: 3, Ns: 1
(pause; LAC's timer started first, so fires first)	
ICRQ       -> Nr: 1, Ns: 2	
(Realizing that it has already seen this packet, the LNS discards the packet and sends a ZLB)	

<- ZLB  
Nr: 3, Nr: 2

(LNS's retransmit timer fires)

<- ICRP  
Nr: 3, Nr: 1

ICCN ->  
Nr: 2, Nr: 3

<- ZLB  
Nr: 4, Nr: 2

## Appendix C: Intellectual Property Notice

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF Secretariat."

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

## Full Copyright Statement

Copyright (C) The Internet Society (1999). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.



