Network Working Group
Internet-Draft
Category: Standards Track
<draft-ietf-l2tpext-l2tp-base-03.txt>

J. Lau
M. Townsley
A. Valencia
G. Zorn
cisco Systems
I. Goyret
Lucent Technologies
G. Pall
Microsoft Corporation
A. Rubens
Nexthop
B. Palter
Redback Networks

June 2002

# Layer Two Tunneling Protocol (Version 3) "L2TPv3"

## Status of this Memo



## Copyright Notice

Abstract

This document describes the Layer Two Tunneling Protocol (L2TP). L2TP tunnels Layer 2 packets across an intervening network in a way that is as transparent as possible to both end users and applications.

Acknowledgments

The basic concept for L2TP and many of its protocol constructs were adopted from L2F [RFC2341] and PPTP [RFC2637]. Authors of these drafts are A. Valencia, M. Littlewood, T. Kolar, K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn.

Danny Mcpherson and Suhail Nanji published the first "L2TP Service Type" draft which defined the use of L2TP for tunneling of multiple L2 payload types. This step led to the eventual creation of this document and the modularization of L2TP and PPP tunneling with L2TP.

The team for splitting RFC 2661 into this base document and the companion PPP document consisted of Ignacio Goyret, Jed Lau, Bill Palter, Mark Townsley, and Madhvi Verma. Skip Booth also provided very helpful review and comment.

Stewart Bryant and Simon Barber provided input for the new L2TPv3 over IP header.

This document was based upon RFC 2661, for which a number of people provided valuable input and effort:

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 RFC 2661.

Thomas Narten provided a great deal of critical review and formatting. He originally wrote the IANA Considerations section.

Dory Leifer made valuable refinements to the protocol definition of L2TP and contributed to the editing of early drafts leading to RFC 2661.

Steve Cobb and Evan Caves redesigned the state machine tables.

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

Contents

## 1. Introduction

The Layer Two Tunneling Protocol (L2TP) provides a dynamic tunneling mechanism for multiple Layer 2 (L2) circuits across a packet-oriented data network. L2TP, as originally defined in RFC 2661, is a standard method for tunneling PPP sessions. L2TP has since been adopted for tunneling a number of other L2 protocols. In order to provide greater modularity, this document describes the base L2TP protocol, independent of the L2 payload that is being tunneled.

The base L2TP protocol consists of (1) the control protocol for dynamic creation, maintenance, and teardown of L2TP sessions, and (2) the L2TP data encapsulation to multiplex and demultiplex L2 data streams between two L2TP peers.

### 1.1 Changes from RFC 2661

Most of the protocol constructs described in this document are carried over from RFC 2661. Changes include clarifications based on years of interoperability and deployment experience as well as modifications to either improve protocol operation or provide a clearer separation from PPP. The intent of these modifications is to achieve a healthy balance between code, interoperability experience with RFC 2661, and a thoughtful and directed evolution of the protocol as it is applied to new tasks.

When the designation between L2TPv2 and L2TPv3 is necessary, L2TP as defined in RFC 2661 will be referred to as "L2TPv2", corresponding to the value in the Version field of an L2TP control message header. (L2F is defined as "version 1".) At times, L2TP as defined in this document will be referred to as "L2TPv3". Otherwise, the acronym "L2TP" will refer to L2TPv3 or L2TP in general.

Notable differences between L2TPv2 and L2TPv3 include the following:

- Separation of all PPP-related AVPs, references, etc., including a portion of the L2TP data header that was specific to the needs of PPP. The PPP-specific constructs are described in a companion document.

- Transition from a 16-bit Session ID and Tunnel ID to a 32-bit Session ID and Control Connection ID, respectively.

Details of these changes and a recommendation for transitioning to L2TPv3 may be found in Section 4.7.

## 1.2 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].

## 1.3 Terminology

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 control connections. This construct is known as the Type-Length-Value (TLV) in some specifications. (See also: Control Connection, Control Message.)

Call (Circuit Up)

The action of transitioning a circuit on an LAC to an "up" or "active" state. A call may be dynamically established through signaling properties (e.g. an incoming or outgoing call through the PSTN) or statically configured (e.g. provisioning a VC on an interface). A call is defined by its properties (e.g. type of call, called number, etc.) and its data traffic. (See also: Circuit, Session, Incoming Call, Outgoing Call, Outgoing Call Request.)

CHAP

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

Circuit

A general term identifying any one of a wide range of L2 connections. A circuit may be virtual in nature (e.g. an ATM PVC or an L2TP session), or it may have direct correlation to a physical layer (e.g. an RS-232 serial line). Circuits may be statically configured with a relatively long-lived uptime, or

dynamically established with some type of control channel governing the establishment, maintenance, and teardown of the circuit. For the purposes of this document, a statically configured circuit is considered to be largely equivalent to a simple dynamic circuit. (See also: Call, Remote System.)

Client

(See Remote System.)

Control Connection

An L2TP control connection is a reliable control channel that is used to establish, maintain, and release individual L2TP sessions as well as the control channel itself. (See also: Control Message, Data Channel.)

Control Message

An L2TP message used by the control connection. (See also: Control Connection.)

Data Message

Message used by the data channel. (See also: Data Channel.)

Data Channel

The channel of L2TP-encapsulated L2 traffic that passes between two LCCEs, utilizing a specific data encapsulation method. L2TP defines one base encapsulation method for L2 traffic, although others may be used as well. (See also: Control Connection, Data Message.)

Dominant LCCE

The LCCE that either solely initiated establishment of a control connection or won the tie-breaker during control connection establishment. (See also: LCCE, Section 5.4.3.)

Incoming Call

The action of receiving a call (circuit up event) on an LAC. The call may have been placed by a remote system (e.g. a phone call over a PSTN), or it may have been triggered by a local event (e.g. interesting traffic routed to a virtual interface). An incoming call that needs to be tunneled (as determined by the LAC) results in the generation of an L2TP ICRQ message. (See also: Call,

Outgoing Call, Outgoing Call Request.)

L2TP Access Concentrator (LAC)

An LCCE that tunnels a circuit (either physically connected or logically connected, as via another L2TP session) to another location using L2TP, without performing any native L2 packet processing on the circuit. The LAC may tunnel to either an LNS or another LAC. (See also: LCCE, LNS.)

L2TP Control Connection Endpoint (LCCE)

One end of an L2TP control connection, either an LAC or an LNS. (See also: LAC, LNS.)

L2TP Network Server (LNS)

An LCCE that logically terminates a tunneled circuit locally and that processes the tunneled traffic as though the circuit were physically connected to the device. The LNS may tunnel to either an LAC or another LNS. (See also: LCCE, LAC.)

Outgoing Call

The action of placing a call on an LAC, typically in response to policy directed by the peer in an Outgoing Call Request message. (See also: Call, Incoming Call, Outgoing Call Request.)

Outgoing Call Request

A request sent to an LAC to place an outgoing call. The request contains specific information for the LAC in placing the call, information that is typically not known a priori by the LAC. (See also: Call, Incoming Call, Outgoing Call.)

Packet-Switched Network (PSN)

A network layer that uses packet-switching technology for data delivery. This layer is principally IP. Other examples include MPLS, FR, and ATM.

Peer

When used in context with L2TP, Peer refers to the far end of an L2TP control connection (i.e. the far LCCE). An LAC's peer may be either an LNS or another LAC. Similarly, an LNS's peer may be either an LAC or another LNS. (See also: LAC, LCCE, LNS.)

Pseudowire (PW)

An emulated circuit as it traverses a PSN. There is one Pseudowire per L2TP Session. (See also: Packet-Switched Network, Session.)

Pseudowire Type

The payload type being carried within an L2TP session. Examples include PPP, Ethernet, and Frame Relay. (See also: Session.)

Remote System

An end-system or router connected by a circuit to an LAC.

Session

An L2TP session is created by a particular L2TP control connection between two LCCEs when a circuit is successfully established. The circuit may either pass through (LAC) or terminate locally (LNS) on the LCCEs, which maintain state for the circuit. There is a one-to-one relationship between established L2TP sessions and their associated circuits. (See also: Circuit, LAC, LCCE, LNS.)

Zero-Length Body (ZLB) Message

A control message with only an L2TP header. ZLB messages are used for explicitly acknowledging packets on the reliable control channel. (See also: Control Message.)

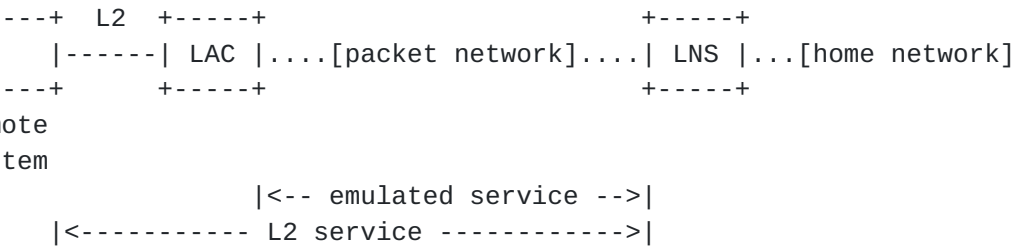
## 2. Topology

L2TP operates between two L2TP Control Connection Endpoints (LCCEs), tunneling circuit traffic across a packet network. An L2TP Network Server (LNS) is an LCCE that decapsulates tunneled L2 traffic and directs it as incoming data towards a virtual L2 interface. In contrast, an L2TP Access Concentrator (LAC) is an LCCE that merely forwards tunneled traffic directly to a circuit (which may even be another L2TP session).

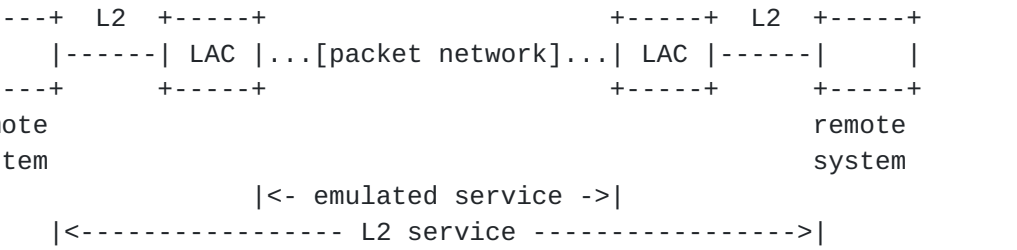
There are three predominant tunneling models in which L2TP operates: LAC-LNS (or vice versa), LAC-LAC, and LNS-LNS. These models are diagrammed below. (Dotted lines designate network connections. Solid lines designate circuit connections.)

Figure 2.0: L2TP Reference Models

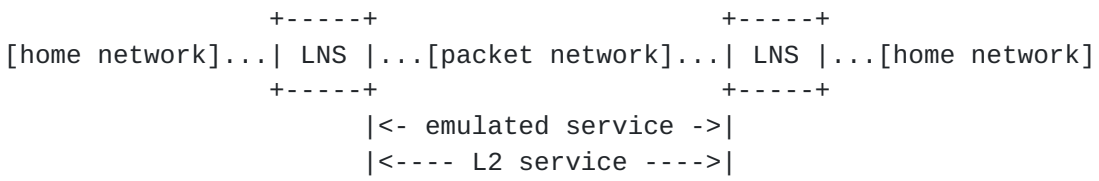
(a) LAC-LNS Reference Model: On one side, the LAC receives traffic from an L2 circuit, which it forwards via L2TP across an IP or other packet-based network. On the other side, an LNS logically terminates the L2 circuit locally and routes traffic (at Layer 3) to the home network. The action of session establishment may be driven by the LAC (perhaps as an incoming call) or the LNS (perhaps as an outgoing call). This model typically has, but does not require, a clear initiator and responder.




(b) LAC-LAC Reference Model: In this model, both LCCEs are LACs. Each LAC forwards circuit traffic from the remote system to the peer LAC using L2TP, and vice versa. A LAC does not perform any native handling of the tunneled L2 frame, and thus, does not utilize a virtual L2 interface. Rather, a LAC acts as a simple cross-connect between a circuit and an L2TP session. This model typically involves symmetric establishment; that is, either side of the connection may initiate a session at any time (or perhaps simultaneously).




(c) LNS-LNS Reference Model: This model has two LNSs as the LCCEs. Rather than forwarding traffic directly over a circuit, each LNS logically terminates the tunneled L2TP session locally. In this manner, both sides have virtual interfaces associated with each L2TP session. A user-level, traffic-generated, or signaled event typically drives session establishment from one side of the tunnel. Also known as "voluntary tunneling" (see [RFC2809]).




Note: If an LNS initiates session establishment due to an event (generally user-driven), the LNS is sometimes referred to as a "LAC Client" as defined in [RFC2661].

## 3. Protocol Overview

L2TP utilizes two types of messages, control messages and data messages. Control messages are used in the establishment, maintenance, and clearing of control connections and sessions. These messages utilize a reliable control channel within L2TP to guarantee delivery (see Section 4.2 for details). Data messages are used to encapsulate the L2 traffic being carried over the L2TP session. Unlike control messages, data messages are not retransmitted when packet loss occurs.

While both the L2TP control channel and the L2TP data channel are defined strictly in this document, the L2TP data channel MAY be substituted with a different L2 tunneling encapsulation whose format can negotiated by the L2TP control connection. Furthermore, the L2TP data channel MAY be used without the control channel, if so desired. However, it is strongly recommended that such practice be limited to relatively small-scale deployments or deployments in which some other form of automatic control information distribution is employed.

Figure 3.0: L2TPv3 Structure

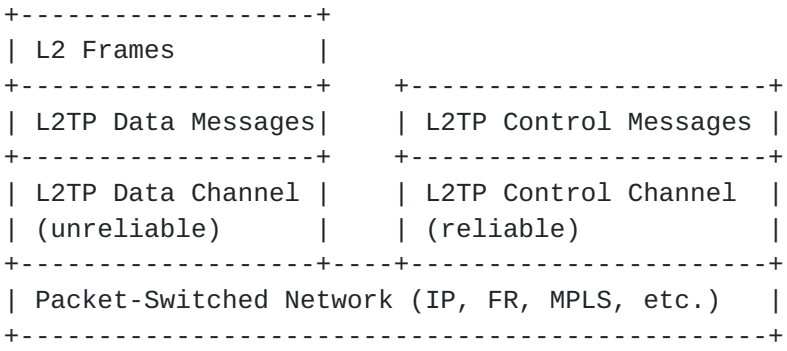



Figure 3.0 depicts the relationship of control messages and data messages over the L2TP control and data channels, respectively. Data messages are passed over an unreliable data channel, encapsulated by an L2TP header, and sent over a Packet-Switched Network (PSN) such as IP, UDP, Frame Relay, ATM, MPLS, etc. Control messages are sent over

a reliable L2TP control channel, which operates over the same PSN.

The necessary setup for tunneling a session with L2TP consists of two steps: (1) Establishing the control connection, if required, and (2) establishing a session as triggered by an incoming call or outgoing call. An L2TP session MUST be established before L2TP can begin to forward session frames. Multiple sessions may be bound to a single control connection, and multiple control connections may exist between the same two LCCEs.

## 3.1 Control Message Types

The Message Type AVP (see Section 5.4.1) defines the specific type of control message being sent.

This document defines the following control message types (see Sections 6.1 through 6.13 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
```

Link Status Change Reporting

```
   16 (SLI)      Set-Link-Info
```

## 3.2 L2TP Header Formats

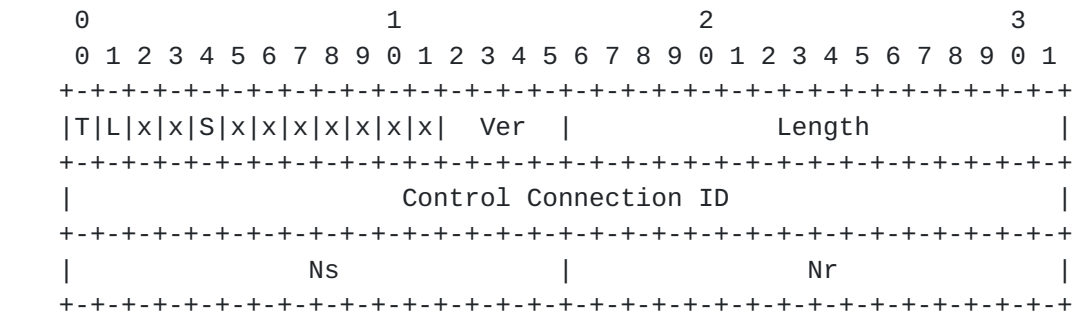
This section defines header formats for L2TP control messages and L2TP data messages. All values are placed into their respective fields and sent in network order (high-order octets first).

### 3.2.1 L2TP Control Message Header

The L2TP control message header provides information for the reliable transport of messages that govern the establishment, maintenance, and teardown of L2TP sessions. By default, control messages are sent over the underlying media in-band with L2TP data messages. As such, L2TP also includes a default method (borrowing from RFC 2661 by utilizing the high bit of the first octet in the L2TP header) that may be used to distinguish L2TP control messages from L2TP data messages. Other methods for distinguishing control and data messages MAY be utilized for specific media (e.g. L2TP over IP, as defined in Section 4.1.1).

The L2TP control message header is formatted as follows:

Figure 3.2.1: L2TP Control Message Header




The T bit MUST be set to 1, indicating that this is a control message.

The L and S bits MUST be set to 1, indicating that the Length field and sequence numbers are present.

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

The Ver field indicates the version of the L2TP control message header described in this document. On sending, this field MUST be set to 3 for all messages (unless operating in an environment with L2TPv2 [RFC2661] and/or L2F [RFC2341], see Section 4.1 for details).

The Length field indicates the total length of the message in octets, always calculated from the start of the control message header itself (beginning with the T bit).

The Control Connection ID field contains the identifier for the control connection. L2TP control connections are named by identifiers that have local significance only. That is, the same control connection will be given unique Control Connection IDs by each LCCE from within each endpoint's own Control Connection ID number space. As such, the Control Connection ID in each message is that of the intended recipient, not the sender. Non-zero Control Connection IDs are selected and exchanged as Assigned Control Connection ID AVPs during the creation of a control connection.

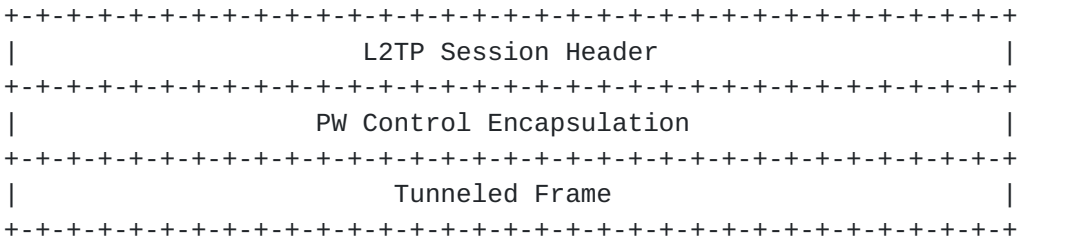
Ns indicates the sequence number for this control message, beginning at zero and incrementing by one (modulo 2**16) for each message sent. See Section 4.2 for more information on using this field.

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). See Section 4.2 for more information on using this field.

### 3.2.2 L2TP Data Message

In general, an L2TP data message consists of a (1) Session Header, (2) an optional PW Control Encapsulation, and (3) the Tunneled L2 Frame, as depicted below.

Figure 3.2.2: L2TP Data Message




The L2TP Session Header is specific to the PSN over which the L2TP traffic is delivered. The Session Header SHOULD provide (1) a method of distinguishing traffic among multiple L2TP data sessions and (2) a method of distinguishing data messages from control messages (assuming the messages are received in-band).

Each type of PSN MUST define its own session header, clearly identifying the format of the header and parameters necessary to setup the session. Section 4.1 defines two session headers, one for

transport over UDP and one for transport over IP.

The PW Control Encapsulation is an intermediary layer between the L2TP session header and the start of the tunneled frame. It SHOULD contain control fields that are used to facilitate the tunneling of each frames (e.g. sequence numbers). The default PW control encapsulation for L2TPv3 is defined in Section 4.6.

The Tunneled Frame consists of PW data traffic, including any necessary L2 framing as defined in the payload-specific companion documents.

## 3.3 Control Connection Management

The L2TP Control Connection handles dynamic establishment, teardown, and maintenance of the L2TP sessions and of the control connection itself. The reliable delivery of control messages is described in Section 4.2.

This section describes the typical control connection establishment and teardown exchanges. It is important to note that, in the diagrams that follow, the reliable control message delivery mechanism exists independently of the L2TP state machine. For instance, ZLB ACKs may be sent after any of the control messages indicated in the exchanges below if an acknowledgment is not piggybacked on a later control message.

### 3.3.1 Control Connection Establishment

Establishment of the control connection involves an exchange of AVPs that identifies the peer and its capabilities.

A three-message exchange is used to establish the control connection. The following is a typical message exchange:

```
LCCE A      LCCE B
------      ------
SCCRQ ->
            <- SCCRP
SCCCN ->
```

### 3.3.2 Control Connection Teardown

Control connection teardown may be initiated by either LCCE and is accomplished by sending a single StopCCN control message. As part of the reliable control message delivery mechanism, the recipient 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 at least 31 seconds (see Section 4.2). The following is an example of a typical control message exchange:

```
LCCE A      LCCE B
------      ------
StopCCN ->
(Clean up)

            (Wait)
            (Clean up)
```

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

## 3.4 Session Management

After successful control connection establishment, individual sessions may be created. Each session corresponds to a single data stream between the two LCCEs. This section describes the typical call establishment and teardown exchanges.

### 3.4.1 Session Establishment for an Incoming Call

A three-message exchange is used to establish the session. The following is a typical sequence of events:

```
LCCE A      LCCE B
------      ------
(Call
 Detected)

ICRQ ->
         <- ICRP
ICCN ->
```

### 3.4.2 Session Establishment for an Outgoing Call

A three-message exchange is used to set up the session. The following is a typical sequence of events:

```
LCCE A      LCCE B
------      ------
         <- OCRQ
OCRP ->

(Perform
 Call
 Operation)

OCCN ->
```

### 3.4.3 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). The following is an example of a typical control message exchange:

```
LCCE A      LCCE B
------      ------
CDN ->
(Clean up)

            (Clean up)
```

## 4. Protocol Operation

This section addresses various operational issues in both the control connection and data channel of L2TP.

### 4.1 L2TP Over Specific Packet-Switched Networks (PSN)

L2TP is designed to allow operation over a variety of PSNs. The L2TP Session Header encapsulation MAY vary for a given PSN.

This document describes the standard method for operation of L2TP over IPv4 networks. There are two modes described, L2TP over IP (Section 4.1.1) and L2TP over UDP (Section 4.1.2). L2TPv3 implementations MUST support L2TP over IP and SHOULD support L2TP over UDP for better NAT and FW traversal, integration with IPsec [RFC3193], and easier migration from L2TPv2.

L2TP over other PSNs may be defined, but the specifics are outside the scope of this document. Examples of L2TPv2 over other PSNs include [RFC3070] and [L2TPAAL5].

The following field definitions are defined for use in all L2TP

Session Header encapsulations.

Session ID

A 32-bit field containing a non-zero identifier for a session. L2TP sessions are named by identifiers that have local significance only. That is, the same logical session will be given different Session IDs by each end of the control connection for the life of the session. When the L2TP control connection is used for session establishment, Session IDs are selected and exchanged as Local Session ID AVPs during the creation of a session.

Cookie

The optional Cookie field contains a variable length (maximum 64 bits), longword-aligned value used to check the association of a received data message with the session identified by the Session ID. The Cookie MUST be configured with a random value utilizing all bits in the field. The Cookie provides an additional level of guarantee, beyond the Session ID lookup, that a data message has been directed to the proper session. A well-chosen Cookie may prevent inadvertent misdirection of stray packets with recently reused Session IDs, Session IDs subject to packet corruption, etc.

When the L2TP control connection is used for session establishment, random Cookie values are selected and exchanged as Assigned Cookie AVPs during the creation of a session. The maximum size of the Cookie field is 64 bits.

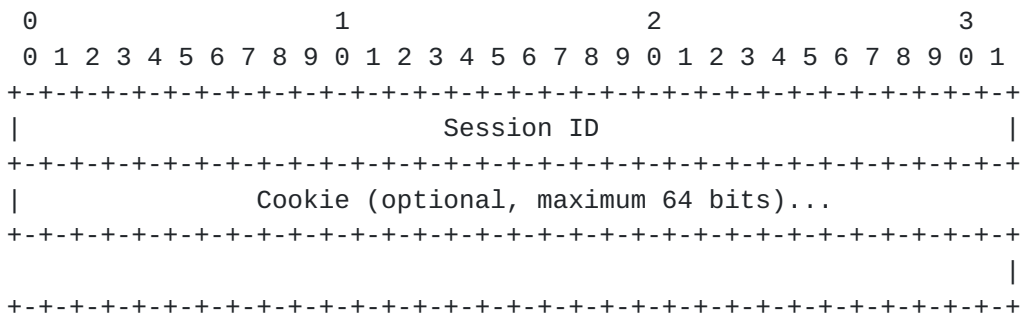
### 4.1.1 L2TP over IP

L2TP over IP utilizes the IANA assigned IP protocol ID 115.

#### 4.1.1.1 L2TP over IP Session Header

Unlike L2TP over UDP, the L2TPv3 session header over IP is free of any restrictions imposed by coexistence with L2TPv2 and L2F. As such, the header format has been redesigned to optimize packet processing. The following session header format is utilized when operating L2TPv3 over IP:

Figure 4.1.1.1: L2TPv3 over IP Session Header




The Session ID and Cookie fields are as defined in Section 4.1. The Session ID of zero is reserved for use by L2TP control messages (see Section 4.1.1.2).

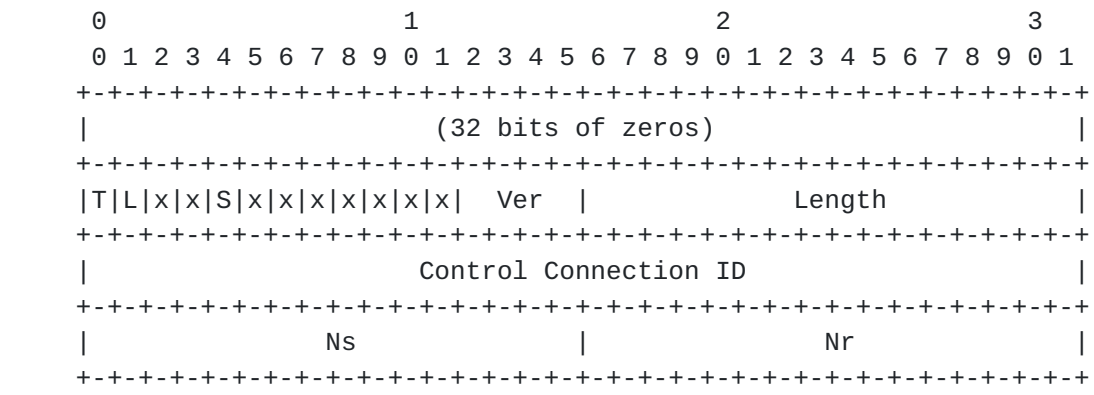
It should be noted that the absence of the Version and Flags fields, which are present in L2TP over UDP, prevents straightforward version extensibility for data messages. However, given the freedom of setting the first 32 bits in the data message header (i.e. the Session ID field), an acceptable workaround to this limitation can be devised if an extension to the demultiplexing capabilities of L2TP is ever in need of further revision. In contrast, the control message header still retains all version checking ability.

### 4.1.1.2 L2TP Control and Data Traffic over IP

As shown in Section 4.1.1.1, there are no Version and Flags fields in the L2TP Session Header over IP. Specifically, the T bit does not exist to distinguish control packets and data packets. Instead, all control packets are sent over the reserved session ID of 0. It is presumed that this method is more efficient -- both in header size for data packets and in processing speed for distinguishing control messages -- than checking for the presence of certain bits.

The entire control message header over IP, including the zero session ID, appears as follows:

Figure 4.1.1.2: L2TPv3 over IP Control Message Header




Named fields are as defined in Section 3.2.1. Note that the Length field is still calculated from the beginning of the control message header, beginning with the T bit. It does NOT include the "(32 bits of zeros)" depicted above.

### 4.1.2 L2TP over UDP

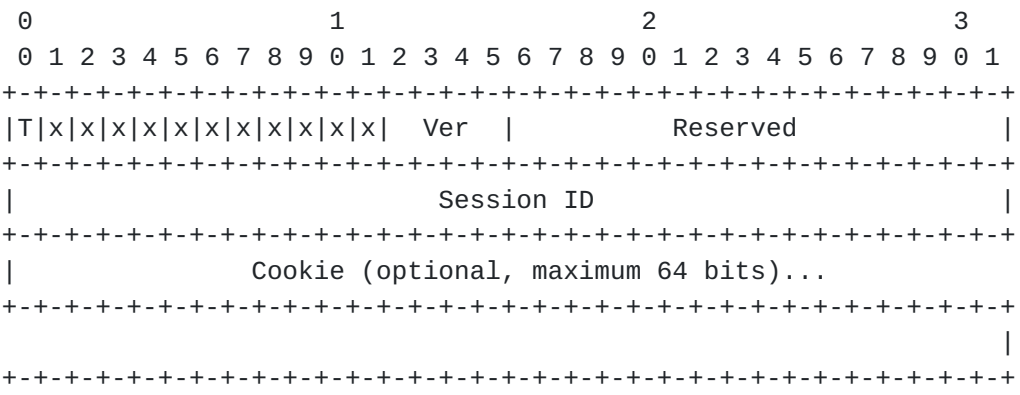
L2TPv3 over UDP must consider other L2 tunneling protocols that may be operating in the same environment, including L2TPv2 [RFC2661] and L2F [RFC2341].

While there are efficiencies gained by running L2TP directly over IP, there are possible side effects as well. For instance, L2TP over IP is not as NAT-friendly as L2TP over UDP. Also, control messages transmitted over IP are not protected by a network-layer checksum as they are with UDP.

#### 4.1.2.1 L2TP over UDP Session Header

The following session header format is utilized when operating L2TPv3 over UDP:

Figure 4.1.2.1: L2TPv3 over UDP Session Header




The T bit MUST be set to 0, indicating that this is a data message.

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

The Ver field MUST be set to 3, indicating an L2TPv3 message.

The Session ID and Cookie fields are as defined in Section 4.1.

### 4.1.2.2 L2TP over UDP Port Selection

L2TPv3 utilizes the same UDP port selection method as defined in L2TPv2 [RFC2661].

When negotiating a control connection over UDP, control messages first must be sent as UDP datagrams using the registered UDP port 1701 [RFC1700]. The initiator of an L2TP control connection 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.

Any subsequent traffic associated with this control connection (either control traffic or data traffic from a session established through this control connection) must use these same UDP ports. This method has some inefficiencies with regard to packet processing. However, it is the most NAT-friendly method since there is only one entry in the NAT table to be kept valid, and the control connection can provide a keepalive to ensure that the NAT entry remains valid. Also, firewalls can be configured to pass all control and data traffic with a single entry rather than separate entries for control and for data.

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 control connection, i.e., 1701) may make it more difficult for L2TP to traverse some NAT devices. Implementations should consider the potential implication of this capability before choosing an arbitrary source port. Any NAT device that can pass TFTP traffic should be able to pass L2TP UDP traffic since both protocols employ similar policies with regard to UDP port selection.

#### 4.1.2.3 UDP Checksum

UDP checksums MUST be enabled for control messages and MAY be enabled for data messages. It should be noted, however, that enabling checksums on data packets may significantly increase packet processing burden.

### 4.1.3 IP Fragmentation Issues

IP fragmentation may occur as the L2TP packet travels over the IP substrate. L2TP makes no special efforts defined in this document to optimize this.

## 4.2 Reliable Delivery of Control Messages

L2TP provides a lower level reliable delivery service for all control messages. The Nr and Ns fields of the control message header (see Section 3.2.1) belong to this delivery mechanism. The upper level functions of L2TP are not concerned with retransmission or ordering of control messages. The reliable control messaging mechanism is a sliding window mechanism that provides control message retransmission and congestion control. Each peer maintains separate sequence number state for each control connection.

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 delivery mechanism. This acknowledgment may either piggybacked on a message in queue or sent explicitly via a

ZLB ACK.

All control messages take up one slot in the control message sequence number space, except the ZLB acknowledgment. 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), the Nr of the next message sent is not updated by the Ns of the ZLB. As a precaution, Nr should be sanity-checked before flushing the retransmit queue. For instance, if the Nr received in a control message is greater than the last Ns sent plus 1 modulo 65536, the control message is clearly invalid.

The reliable delivery mechanism 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. Alternatively, they may be discarded, thus requiring a retransmission by the peer. When dropping out of order control packets, Nr MAY be updated before the packet is discarded.

Each control connection 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 acknowledgment, 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 control connection and all associated sessions MUST be cleared.

When a control connection 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 and B. Suppose A specifies a Receive Window Size AVP with a value of N in the SCCRQ or SCCRP message. B is now allowed to have up to N outstanding control messages. Once N messages have been sent, B must wait for an acknowledgment from A that advances the window before sending new control messages. An implementation may support a receive window of only 1 (e.g. 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 (i.e. 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. A recommended procedure is described in Appendix A.

A peer MUST NOT withhold acknowledgment of messages as a technique for flow control of 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 actions.

In addition, a peer MUST NOT withhold acknowledgment of messages in order to maintain state in the L2TP state machine. Conversely, the L2TP state machine MUST be capable of maintaining state if a ZLB ACK is received in response to a control message. However, determining when a state should no longer be maintained (e.g. how long to wait in wait-reply state for an ICRP from the peer) before destroying a session or control connection is an issue that is left to each implementation.

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

## 4.3 Control Connection Authentication

L2TP incorporates a simple, optional, CHAP-like [RFC1994] authentication system for each LCCE during control connection establishment. If an LAC or LNS wishes to authenticate the identity of its peer, 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 SCCCN, respectively. If the expected response received from a peer does not match, establishment of the control connection MUST be disallowed.

To participate in LCCE authentication, a single shared secret MUST

exist between the two LCCEs. This is the same shared secret used for AVP hiding (see Section 5.3). See Section 5.4.3 for details on construction of the Challenge and Response AVPs.

## 4.4 Keepalive (Hello)

A keepalive mechanism is employed by L2TP to detect loss of connectivity between a pair of LCCEs. This detection is accomplished by injecting Hello control messages (see Section 6.5) after a specified period of time has elapsed since the last data message or control message was received on an L2TP session or control connection, respectively. As with any other control message, if the Hello message is not reliably delivered, the sending LCCE declares that the control connection is down and resets its state for the control connection. This behavior ensures that a connectivity failure between the LCCEs is detected independently by each end of a control connection.

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 acknowledgment information.

If the control channel is operated in-band with data traffic over the PSN, this single mechanism can be used to infer basic data connectivity between a pair of LCCEs for all sessions associated with the control connection.

Keepalives for the control connection 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. An LCCE sending Hello messages across multiple control connections between the same LCCE endpoints SHOULD employ a jittered timer mechanism.

## 4.5 Forwarding Session Data Frames

Once session establishment is complete, circuit frames are received at an LCCE, encapsulated in L2TP (with appropriate attention to framing as described in documents for the particular pseudowire type), and forwarded over the appropriate session. For every outgoing data message, the sender places the identifier specified in the Local Session ID AVP (received from peer during session establishment) in the Session ID field of the L2TP data header. In this manner, session frames are multiplexed and demultiplexed between a given pair of LCCEs. Multiple control connections may exist

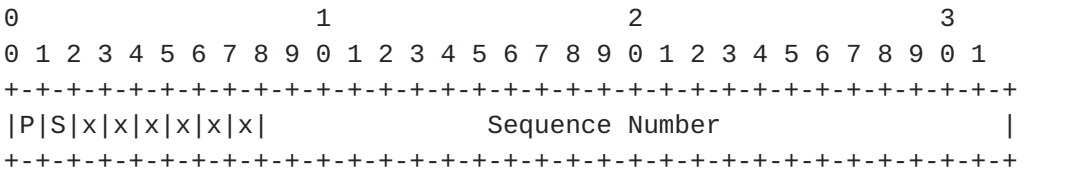
between a given pair of LCCEs, and multiple sessions may be associated with a given control connection.

The peer LCCE receiving the L2TP data packet identifies the session with which the packet is associated by the Session ID in the data packet's header. The LCCE then checks the Cookie field in the data packet against the Cookie value received in the Assigned Cookie AVP during session establishment. Any received data packets that contain invalid Session IDs or associated Cookie values MUST be dropped. Finally, the LCCE either processes the encapsulated session frame locally (i.e. as an LNS) or forwards the frame to a circuit (i.e. as an LAC).

## 4.6 Default PW Control Encapsulation

This document defines a default PW control encapsulation (see Section 3.2.2) format that a pseudowire may use for features such as basic sequencing support, marking of packets with a single high-priority bit, or other general PW-specific per-packet control operations. The default control encapsulation SHOULD be used by a given PW type to support these features if it is adequate, and its presence is requested by a peer during session negotiation. Alternative PW control encapsulations MAY be defined (e.g. an encapsulation with a larger Sequence Number field) and identified for use via the PW Control Encapsulation Type AVP.

Figure 4.6: Default PW Control Encapsulation Format




The P (Priority) bit is used to identify a data packet that should be dropped only as a last resort after being received by an L2TP peer. This bit should be set to 1 for any traffic that should be given higher priority than other data traffic in a congested environment. For example, end-to-end L2 keepalive packets (e.g. LCP keepalives) or other control packets vital to the life of the circuit may need special handling by an LCCE upon receipt. This is not a replacement for, or to be used as, a per-hop QoS method of any sort. It is only to be used by the L2TP receiving node to prioritize incoming traffic.

The S (Sequence) bit is set to 1 when the Sequence Number contains a valid number for this sequenced frame. If the S bit is set to zero, the Sequence Number contents are undefined and MUST be ignored by the receiver.

The Sequence Number field contains a free-running counter of 2^24 sequence numbers. If the number in this field is valid, the S bit MUST be set to 1. The Sequence Number begins at zero, which is a valid sequence number. (In this way, implementations inserting sequence numbers do not have to "skip" zero when incrementing.) 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 (2^23-1) values, inclusive.

### 4.6.1 Sequencing Data Packets

The Sequence Number field may be used to detect lost packets and/or restore the original sequence of packets that may have been reordered during traversal of the packet network.

When L2 frames are carried over an L2TP-over-IP or L2TP-over-UDP/IP data channel, this part of the link has the characteristic of being able to reorder or silently drop packets. Reordering may break some non-IP protocols or L2 control traffic being carried by the link. Silent dropping of packets may break protocols that assume per-packet indication of error, such as TCP header compression. The sequence dependency characteristics of individual protocols are outside the scope of this document.

If any protocol being transported by over L2TP data channels cannot tolerate misordering, sequencing may be enabled on some or all packets by using the S bit and Sequence Number field defined in the default PW control encapsulation (see Section 4.6). For a given L2TP session, each LCCE is responsible for communicating to its peer the level of sequencing support that it requires of data packets that it receives. Mechanisms to advertise this information during session negotiation are provided (see, in particular, the Data Sequencing AVP in Section 5.4.4). PW-specific documents MAY place greater constraints on sequence number enforcement than those defined here.

## 4.7 L2TPv2/v3 Interoperability and Migration

L2TPv2 and L2TPv3 environments should be able to coexist while a migration to L2TPv3 is made. Migration issues are discussed for each media type in this section. Most issues apply only to implementations that require both L2TPv2 and L2TPv3 operation. However, even L2TPv3-only implementations must be mindful of these issues in order to interoperate with implementations that support both versions.

### 4.7.1 L2TPv3 over IP

L2TPv3 implementations running strictly over IP with no desire to interoperate with L2TPv2 implementations may safely disregard most migration issues from L2TPv2. All control messages and data messages are sent as described in this document.

An L2TP implementation may first attempt to operate in L2TPv3 over IP mode, then fall back to L2TPv2 (over UDP) if L2TPv3 over IP is unavailable. It does so by first sending an L2TPv3-formatted SCCRQ over IP to try to initiate an L2TPv3 control connection. If the SCCRQ elicits no response, the implementation may fall back to L2TPv2 operation, as defined in [RFC2661]. Fallback to L2TPv2 should be seamless and occur automatically. (See Section 4.7.3 for further details.)

### 4.7.2 L2TPv3 over UDP

In order to allow simultaneous operation with L2TPv2, L2TPv3 uses the same UDP port (port 1701) as L2TPv2 and shares the first two octets of header format (via the session header) with L2TPv2. Furthermore, though the control message and data message headers have changed, an LCCE sends an SCCRQ that looks enough like an L2TPv2 SCCRQ to be accepted by both L2TPv2 and L2TPv3 implementations. If the response to the SCCRQ is a properly formatted L2TPv3 message, then operation can continue as described in this document for an L2TPv3 implementation. If the response is a properly formatted L2TPv2 message, then an L2TPv2 mode of operation must be adopted.

### 4.7.3 Automatic L2TPv2 Fallback

When running over UDP, an implementation may detect whether a peer is L2TPv3-capable by sending an L2TPv3-formatted SCCRQ. The SCCRQ is sent with the Ver field set to 2, and any L2TPv3-specific AVPs within the message are sent without setting the M bit on each AVP (so that they may be ignored by an L2TPv2 implementation). Note that, in both L2TPv2 and L2TPv3, the value contained in the space of the control message header utilized by the 32-bit Control Connection ID (16-bit Tunnel ID and 16-bit Session ID in L2TPv2) is always 0 for an SCCRQ, a key feature for this capability.

If the peer implementation is an L2TPv3-capable implementation, a control message with Ver set to 3 and corresponding header and message format will be sent in response to the SCCRQ. Operation may then continue as L2TPv3. If a message is received with Ver set to 2, one may assume that the peer implementation is L2TPv2-only and fall back to L2TPv2 mode if local policy and capability permits.

The auto-detection mode requires that an L2TPv3-only implementation be liberal in its acceptance of SCCRQ control messages with the Ver field set to 2. Thus, an L2TPv3 over UDP implementation MUST allow receipt of an SCCRQ with Ver field of 2 or Ver field of 3.

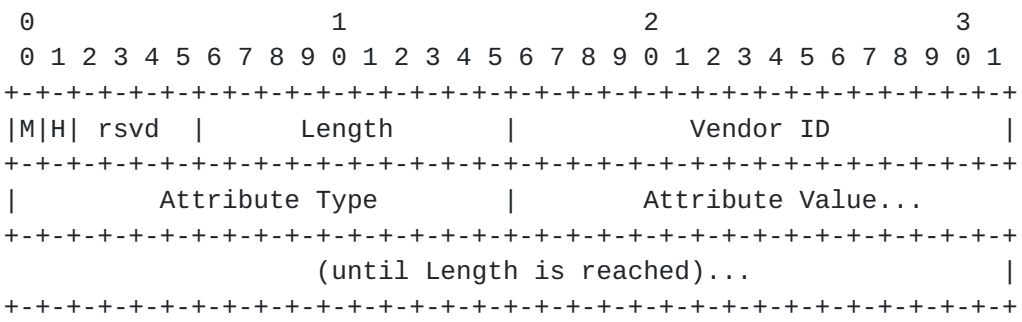
## 5. Control Message Attribute Value Pairs

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) for the remainder of this document.

### 5.1 AVP Format

Each AVP is encoded as follows:

Figure 5.1: AVP Format




The first six bits comprise a bit mask that describes the general attributes of the AVP. Two bits are defined in this document; the remaining bits 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 that receives an unrecognized or malformed AVP. The M bit of a given AVP should only be checked if the AVP is unrecognized or malformed. If the M bit is set on an unrecognized or malformed AVP in a control message associated with a particular session, the session MUST be terminated. If the M bit is set on an unrecognized or malformed AVP within a control message associated with a control connection, the control connection (and all sessions bound to the control connection) 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 5.3 describes the procedure for performing AVP hiding.

Length: Encodes the number of octets (including the Overall Length and bit mask 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 its own L2TP extensions can use its own Vendor ID along with private Attribute values, guaranteeing that they will not collide with any other vendor's extensions or 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.

## 5.2 Mandatory AVPs

Receipt of an unrecognized or malformed AVP that has the M bit set is catastrophic to the session or control connection with which it is associated. Thus, the M bit should only be defined for AVPs that are absolutely crucial to proper operation of the session or control connection. Furthermore, in the case in which the LAC or LNS receives an unknown AVP with the M bit set and shuts down the session or control connection 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 consequence is intended.

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

On the receiving side, the recipient of a control message should only check the M bit of an AVP when the AVP is determined to be unrecognized or malformed. The M bit should not be checked for a recognized and well-formatted AVP. This rule prevents the possibility of a valid AVP resulting in a session or control connection teardown simply because its M bit was set to a value that was unexpected by the receiving LCCE.

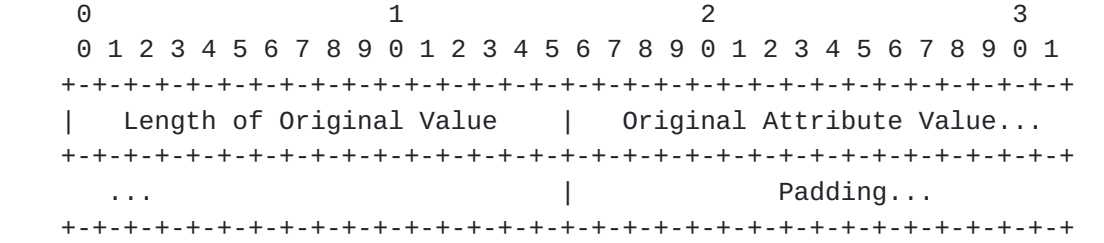
## 5.3 Hiding of AVP Attribute Values

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 (1) a shared secret exists between the LCCEs and (2) LCCE authentication has completed. The shared secret is the same secret that is used for LCCE authentication (see Section 4.3). Hidden values MUST NOT be unhidden until after LCCE authentication has completed successfully (perhaps requiring the hidden value to be stored until after receipt of additional setup messages). To do otherwise runs the risk of AVP data being utilized without verifying the integrity of the shared secret. 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:

Figure 5.3: Hidden AVP Subformat




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 that 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 (in network byte order) on the concatenation of the following:

+ 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. 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 [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.

```
b1 = MD5(AV + S + RV)   c(1) = p1 xor b1
b2 = MD5(S  + c(1))     c(2) = p2 xor b2
           .                       .
           .                       .
           .                       .
bi = MD5(S  + c(i-1))   c(i) = pi xor bi
```

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.

## 5.4 AVP Summary

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

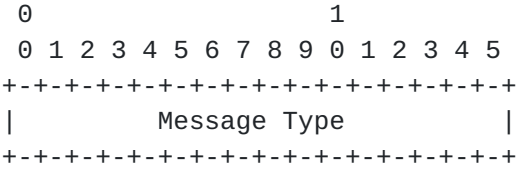
Following the name of the AVP is a list indicating the message types that utilize each AVP. After each AVP title follows a short description of the purpose of the AVP, a detail (including a graphic) of the format for the Attribute Value, and any additional information needed for proper use of the AVP.

### 5.4.1 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:




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 3.2.1). See Section 3.1 for the list of defined control message types and their identifiers.

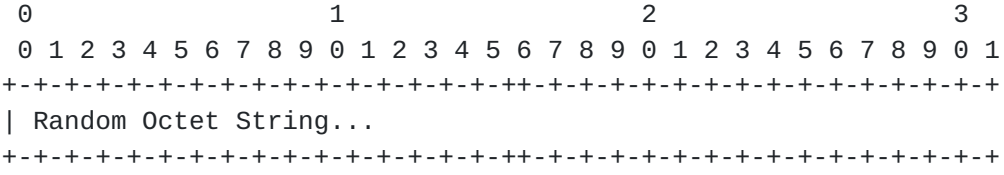
The Mandatory (M) bit within the Message Type AVP has special meaning. Rather than an indication as to whether the AVP itself should be ignored if not recognized or malformed, it is an indication as to whether the control message itself should be ignored. If the M bit is set within the Message Type AVP and the Message Type is unknown to the implementation, the control connection 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.

A vendor-specific control message may be defined by setting the Vendor ID of the Message Type AVP to a value other than the IETF Vendor ID of 0 (see Section 5.1). The Message Type AVP MUST still be the first AVP in the control message.

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:




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 5.3).

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.

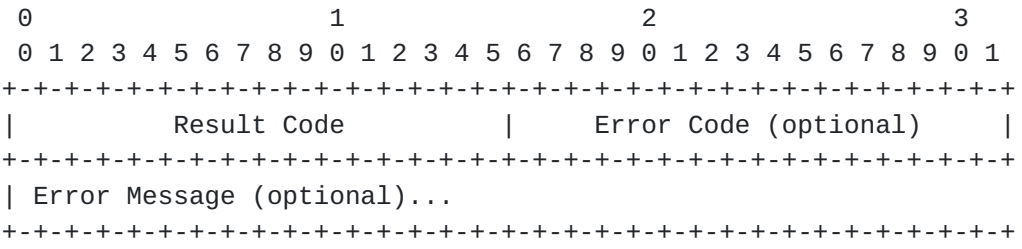
The M bit for this AVP SHOULD be set to 1. This AVP MUST NOT be hidden (the H bit MUST be 0). The Length of this AVP is 6 plus the length of the Random Octet String.

### 5.4.2 Result and Error Codes

Result Code (StopCCN, CDN)

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:




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 is 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 SHOULD 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 as follows:

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.

General Result Code values for the CDN message are as follows:

0 - Reserved.
1 - Session disconnected due to loss of carrier or circuit disconnect.
2 - Session disconnected for the reason indicated in Error Code.
3 - Session disconnected for administrative reasons.
4 - Session establishment failed due to lack of appropriate facilities being available (temporary condition).
5 - Session establishment failed due to lack of appropriate facilities being available (permanent condition).
6 - 11 Reserved (PPP-specific codes defined outside this document).
RC-TBA1 - Session not established due to losing tie-breaker.
RC-TBA2 - Session not established due to unsupported PW type.
RC-TBA3 - Session not established, sequencing required without valid PW control encapsulation.

Additional service-specific Result Codes are defined outside this document.

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 as follows:

0 - No general error.
1 - No control connection exists yet for this pair of LCCEs.
2 - Length is wrong.
3 - One of the field values was out of range.
4 - Insufficient resources to handle this operation now.
5 - Invalid Session ID.
6 - A generic vendor-specific error occurred.
7 - Try another. If initiator is aware of other possible responder destinations, it should try one of them. This can be used to guide an LAC or LNS based on policy.
8 - The session or control connection was shut down due to receipt of an unknown AVP with the M bit set (see Section 5.2). The Error Message SHOULD contain the attribute of the offending AVP in (human-readable) text form.
9 - Try another directed. If an LAC or LNS is aware of other possible destinations, it should inform the initiator of the control connection or session. The Error Message MUST contain a comma-separated list of addresses from which the initiator may

choose. If the L2TP data channel runs over IPv4, then this would be a comma-separated list of IP addresses in the canonical dotted-decimal format (e.g. "10.0.0.1, 10.0.0.2, 10.0.0.3") in the UTF-8 charset using the Default Language [RFC2277]. If there are no servers for the LAC or LNS to suggest, then Error Code 7 should be used. The delimiter between addresses MUST be precisely a single comma and a single space.

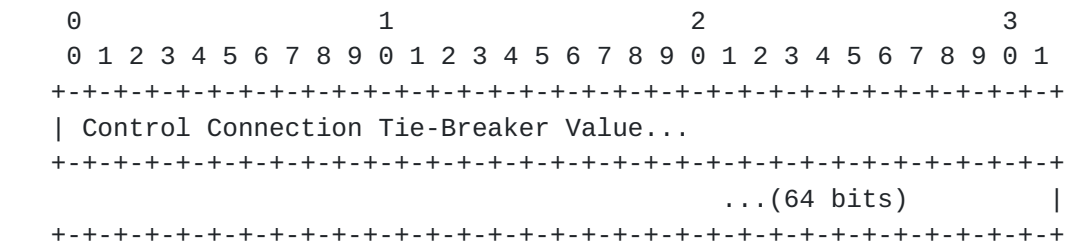
When a General Error Code of 6 is used, additional information about the error SHOULD be included in the Error Message field. Furthermore, a vendor-specific AVP MAY be sent to indicate the problem more precisely.

### 5.4.3 Control Connection Management AVPs

Control Connection Tie-Breaker (SCCRQ)

The Control Connection Tie-Breaker AVP, Attribute Type 5, indicates that the sender desires a single control connection to exist between a given pair of LCCEs.

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




The Control Connection Tie-Breaker Value is an 8-octet random value that is used to choose a single control connection when two LCCEs request a control connection 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 Control Connection Tie-Breaker value with the received one. The lower value "wins", and the "loser" MUST discard its control connection, sending a StopCCN if the SCCRQ that it had sent was acknowledged by the receiving peer. In the case in which a tie-breaker is present on both sides and the value is equal, both sides MUST discard their control connections and restart control connection negotiation with a new, random tie-breaker value.

If a tie-breaker is received and an outstanding SCCRQ has no tie-breaker value, the initiator that included the Control Connection Tie-Breaker AVP "wins". If neither side issues a tie-breaker, then two separate control connections are opened.

Tie-breaker values MUST be random values.

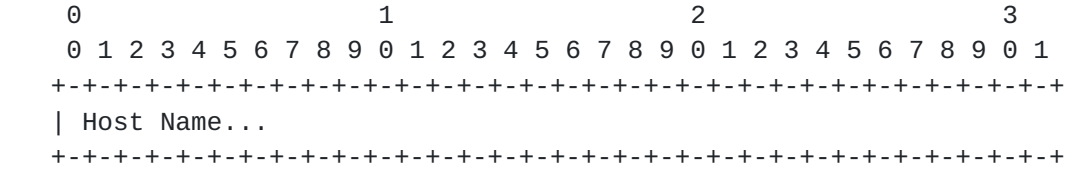
Note that in RFC 2661, this value was referred to as the Tie-Breaker AVP. Here, the AVP serves the same purpose and has the same attribute value and composition.

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

Host Name (SCCRQ, SCCRP)

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:




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

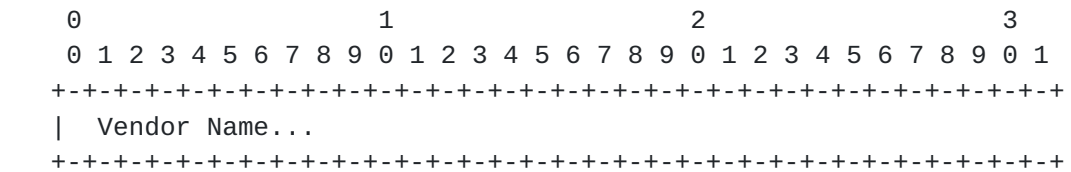
This name should be as broadly unique as possible; for hosts participating in DNS [RFC1034], a hostname with fully qualified domain would be appropriate. The Host Name MAY be used to identify LCCE configuration, including the shared secret for LCCE authentication (if enabled) and any other options defined for the control connection.

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

Vendor Name (SCCRQ, SCCRP)

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:



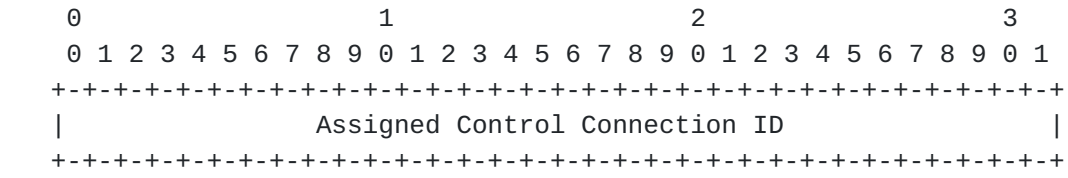
The Vendor Name is the indicated number of octets representing the vendor string. Human-readable text for this AVP MUST be provided in the UTF-8 charset using the Default Language [RFC2277].

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

Assigned Control Connection ID (SCCRQ, SCCRP, StopCCN)

The Assigned Control Connection ID AVP, Attribute Type TBA, encodes the ID being assigned to this control connection by the sender.

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




The Assigned Control Connection ID is a 4-octet non-zero unsigned integer.

The Assigned Control Connection ID AVP establishes the identifier used to multiplex and demultiplex multiple control connections between a pair of LCCEs. Once the Assigned Control Connection ID AVP has been received by an LCCE, the Control Connection ID specified in the AVP MUST be included in the Control Connection ID field of all control packets sent to the peer for the lifetime of the control connection. Before the Assigned Control Connection ID AVP is received from a peer, all control messages MUST be sent to that peer with a Control Connection ID value of 0 in the header. Because a Control Connection ID value of 0 is used in this special manner, the zero value MUST NOT be sent as an Assigned Control Connection ID value.

Under certain circumstances, an LCCE may need to send a StopCCN to a peer without having yet received an Assigned Control Connection ID AVP from the peer (i.e. SCCRQ sent, no SCCRP received yet). In this case, the Assigned Control Connection ID AVP that had been sent to the peer earlier (i.e. in the SCCRQ) MUST be sent as the Assigned Control Connection ID AVP in the StopCCN. This policy allows the peer to try to identify the appropriate control connection via a reverse lookup.

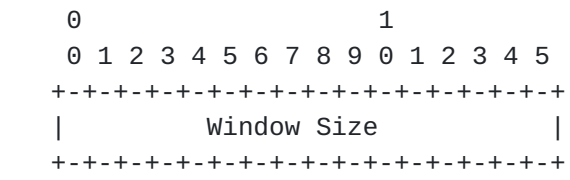
This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 1 (see Section 4.7.3). The Length (before

hiding) of this AVP is 10.

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:




The Window Size is a 2-octet unsigned integer.

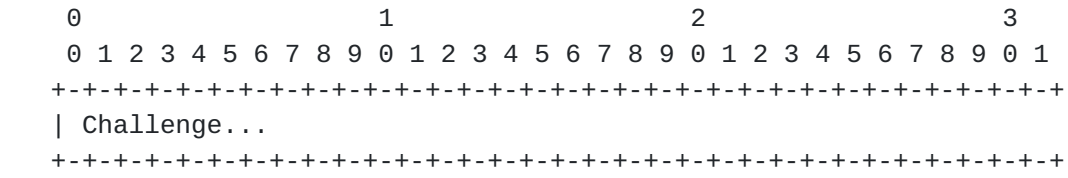
If absent, the peer must assume a Window Size of 4 for its transmit window. The remote peer may send the specified number of control messages before it must wait for an acknowledgment. See Section 4.2 for more information on reliable control message delivery.

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

Challenge (SCCRQ, SCCRP)

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

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




The Challenge is one or more octets of random data.

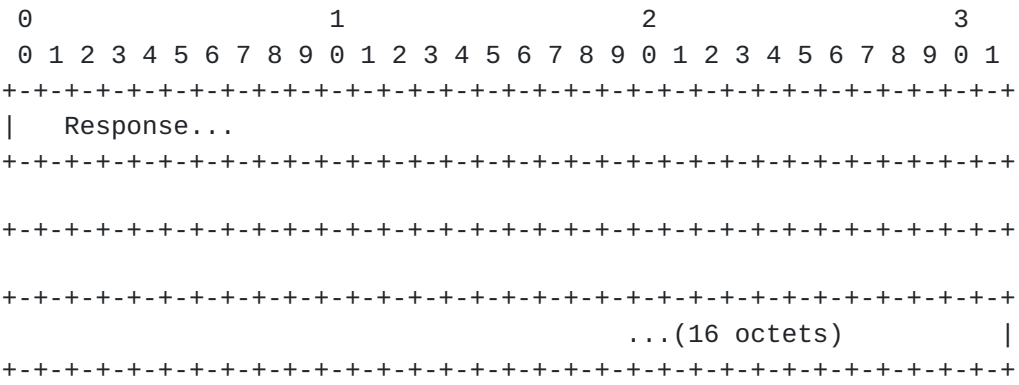
This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 1. The Length (before hiding) of this AVP is 6 plus the length of the Challenge.

Challenge Response (SCCRP, SCCCN)

The Response AVP, Attribute Type 13, provides a response to a

challenge received.

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




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

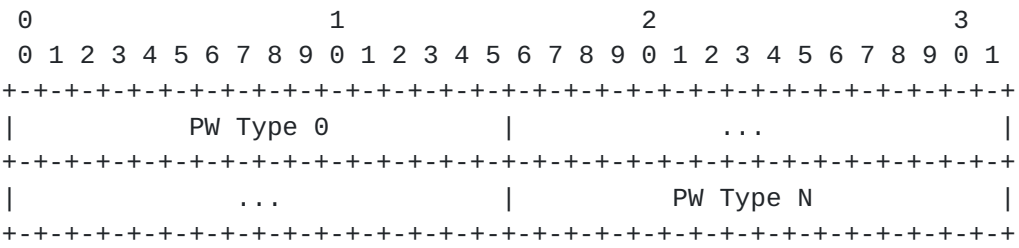
This AVP MUST be present in an SCCRP or SCCCN if a challenge was received in the preceding SCCRQ or SCCRP, respectively. For purposes of the ID value in the CHAP response calculation, the value of the Message Type AVP for this message is used (e.g. 2 for an SCCRP, 3 for an SCCCN).

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

Pseudowire Capabilities List (SCCRQ, SCCRP)

The Pseudowire Capabilities List (PW Capabilities List) AVP, Attribute Type TBA, indicates the L2 payload types the sender can support. The specific payload type of a given session is identified by the Pseudowire Type AVP.

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




Defined PW types that may appear in this list are outside the scope of this document and are managed by IANA. Values 0 to 32767 are

assignable by IETF Consensus [RFC2434]. The remaining values may be assigned on a First Come First Served basis [RFC2434].

If a sender includes a given PW type in the PW Capabilities List AVP, the sender assumes full responsibility for supporting that particular payload, such as any payload-specific AVPs, PW control encapsulation, or control messages that may be defined in the appropriate companion document.

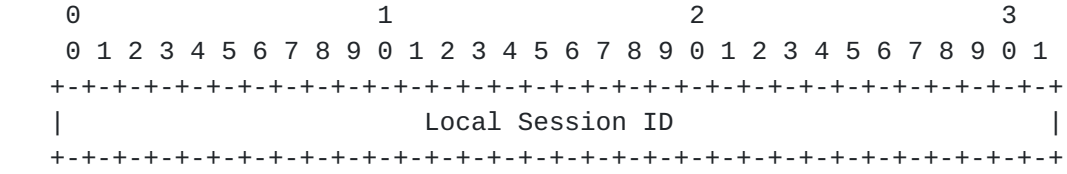
This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 1 (see Section 4.7.3). The Length (before hiding) of this AVP is 8 octets with one PW type specified, plus 2 octets for each additional PW type.

### 5.4.4 Session Management AVPs

Local Session ID (ICRQ, ICRP, ICCN, OCRQ, OCRP, OCCN, CDN, WEN, SLI)

The Local Session ID AVP (analogous to the Assigned Session ID in L2TPv2), Attribute Type TBA, encodes the identifier being assigned to this session by the sender.

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




The Local Session ID is a 4-octet non-zero unsigned integer.

The Local Session ID AVP establishes the identifier used to multiplex and demultiplex both data and control traffic for a given session between two LCCEs. The local LCCE chooses a free value that it sends to the remote LCCE using the Local Session ID AVP. The local LCCE then expects to see this value in the Session ID field of all received data messages for this session. Additionally, for all subsequent session-level control messages received, the local LCCE expects to see this session ID value echoed in the Remote Session ID AVP. On the other side, upon first receiving the Local Session ID AVP in a control message, the remote LCCE MUST use this value for all subsequent messages sent to the local LCCE for this session. The value must appear in the Session ID field in the header of all outgoing data messages for this session, and as the Remote Session ID AVP of all outgoing control messages for this session.

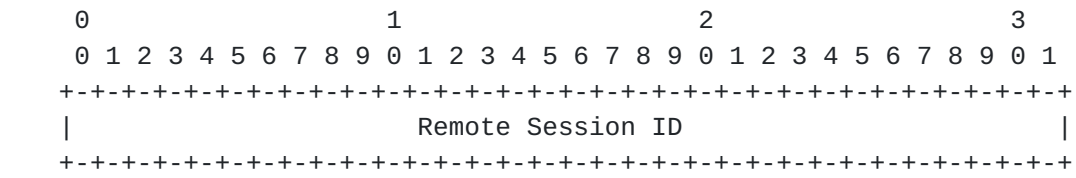
See Section 4.1 for additional information about the Session ID.

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP MUST be 1 for implementations that support only L2TPv3 (see Section 4.7 for L2TPv2 migration issues). The Length (before hiding) of this AVP is 10.

Remote Session ID (ICRQ, ICRP, ICCN, OCRQ, OCRP, OCCN, CDN, WEN, SLI)

The Remote Session ID AVP, Attribute Type TBA, encodes the identifier that was assigned to this session by the peer.

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




The Remote Session ID is a 4-octet non-zero unsigned integer.

The Remote Session ID AVP MUST be present in all session-level control messages. The AVP's value echoes the session identifier advertised by the peer via the Local Session ID AVP. It is the same value that will be used in all transmitted data messages by this side of the session. In most cases, this identifier is sufficient for the peer to look up session-level context for this control message.

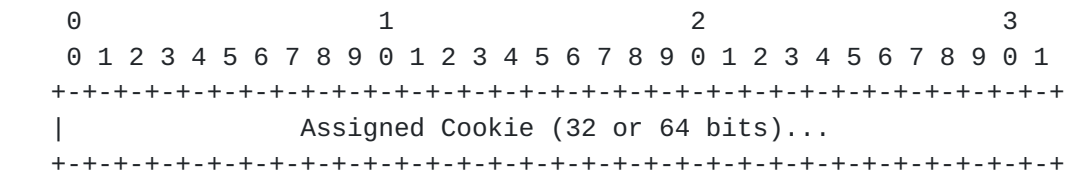
When a session-level control message must be sent to the peer before the Local Session ID AVP has been received from the peer, the value of the Remote Sesson ID AVP MUST be set to zero. Additionally, the Local Session ID AVP (sent in a previous control message for this session) MUST be included in the control message. The peer must then use the Local Session ID AVP to perform a "reverse lookup" to find its session context. Session-level control messages defined in this document that might be subject to a reverse lookup by a receiving peer include the CDN, WEN, and SLI.

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP MUST be 1 for implementations that support only L2TPv3 (see Section 4.7 for L2TPv2 migration issues). The Length (before hiding) of this AVP is 10.

Assigned Cookie (ICRQ, ICRP, OCRQ, OCRP)

The Assigned Cookie AVP, Attribute Type TBA, encodes the Cookie value being assigned to this session by the sender.

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




The Assigned Cookie is a 4-octet or 8-octet random value.

The Assigned Cookie AVP contains the value used to check the association of a received data message with the session identified by the Session ID. All data messages sent to a peer MUST use the Assigned Cookie sent by the peer in this AVP. The value's length (0, 32, or 64 bits) is obtained by the Length of the AVP. A cookie value of zero length serves as positive acknowledgment that the Cookie field should not be present in any data packets sent to this LCCE. The Assigned Cookie AVP MAY not be sent, which has the same effect as sending the AVP to designate a cookie value of zero length.

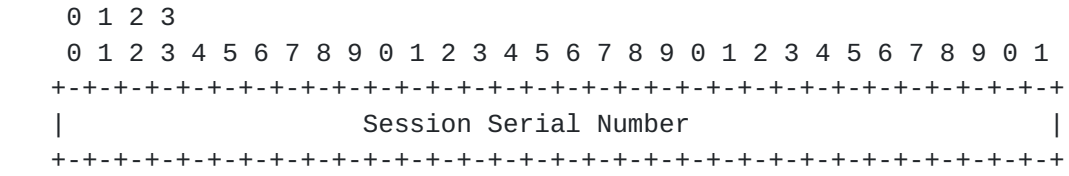
See Section 4.1 for additional information about the Assigned Cookie.

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP MUST be 1 for implementations that support only L2TPv3 (see Section 4.7 for L2TPv2 migration issues). The Length (before hiding) of this AVP may be 6, 10, or 14 octets.

Session Serial Number (ICRQ, OCRQ)

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

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




The Session Serial Number is a 32-bit value.

The Session Serial Number is intended to be an easy reference for administrators on both ends of a control connection to use when investigating session failure problems. Session 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.

Note that in RFC 2661, this value was referred to as the Call Serial

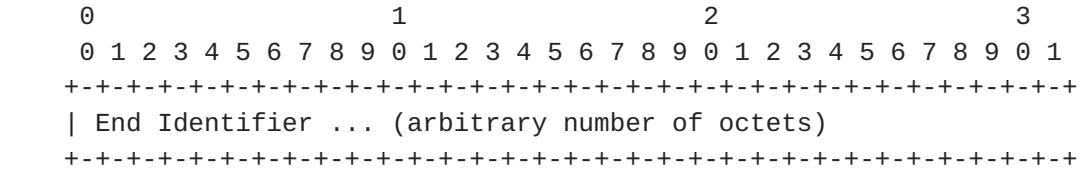
Number AVP. It serves the same purpose and has the same attribute value and composition.

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

End Identifier (ICRQ, OCRQ)

The End Identifier AVP, Attribute Type TBA, encodes an identifier used to associate an attachment circuit with a request for an L2TP session. This AVP allows an LCCE to determine when a session request "tie" has occurred.

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




Use of the End Identifier AVP implies that the session follows the "LAC-LAC" reference model. The End Identifier MUST contain interface, circuit, or remote system information, depending on the circuit that is being tunneled. For example, the field may be a simple 4-octet binary value, a VPN Identifier (as described in [RFC2764]), or an ASCII string. In the simplest case, this value is one that is locally configured, though a directory query MAY be made with this value to obtain additional information about this session request.

A session-level tie is detected if an LCCE receives an ICRQ or OCRQ with an End Identifier AVP whose value and length matches the End Identifier AVP that was just sent in an outgoing ICRQ or OCRQ to the same peer. If the two values match, an LCCE recognizes that a tie exists (i.e. both LCCEs are attempting to establish sessions for the same circuit). The tie is broken by the dominant LCCE, as determined by the Session Tie-Breaker AVP.

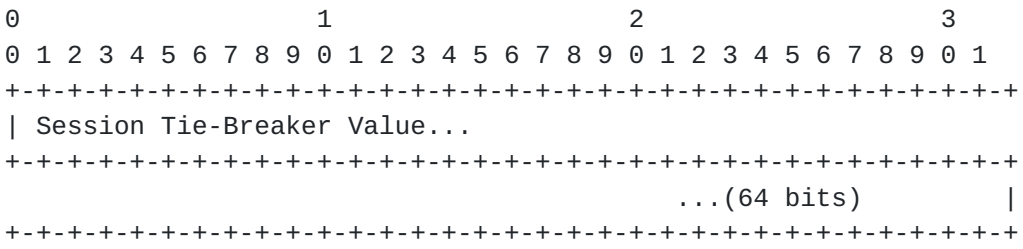
This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 0. The Length (before hiding) of this AVP is 6 plus the length of the End Identifier value.

Session Tie-Breaker (ICRQ, OCRQ)

The Session Tie-Breaker AVP, Attribute Type TBD, is used to break ties when two peers concurrently attempt to establish a session for

the same circuit.

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




The Session Tie-Breaker Value is an 8-octet random value that is used to choose a session when two LCCEs concurrently request a session for the same circuit, as determined by the End Identifier AVP. The recipient of an ICRQ or OCRQ must check to see if an ICRQ or OCRQ, respectively, already has been sent to the peer for the same circuit, and if so, must compare its Session Tie-Breaker Value with the one received. The lower value "wins", and the "loser" MUST send a CDN with result code set to RC-TBA1 (as defined in Section 5.4.2) to tear down the session it instigated. In the case in which a tie-breaker is present on both sides and the value is equal, both sides MUST discard their sessions and restart session negotiation with new random Session Tie-Breaker Values.

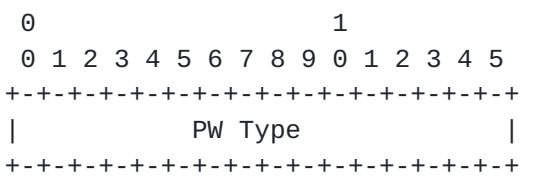
If a tie-breaker is received and an outstanding ICRQ/OCRQ has no tie breaker value, the initiator that included the Session Tie-Breaker AVP "wins". If neither side issues a tie breaker, then both sessions MUST be torn down.

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

Pseudowire Type (ICRQ, OCRQ)

The Pseudowire Type (PW Type) AVP, Attribute Type TBA, indicates the L2 payload type of the packets that will be tunneled using this L2TP session.

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



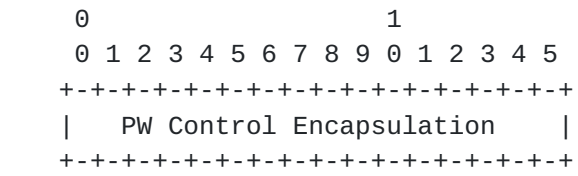

A peer MUST NOT request an incoming or outgoing call with a PW Type

AVP specifying a value not advertised in the PW Capabilities List AVP it received during control connection establishment. Attempts to do so MUST result in the call being rejected via a CDN with the Result Code set to RC-TBA2 (see Section 5.4.2).

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP MUST be 1 for implementations that support only L2TPv3 (see Section 4.7 for L2TPv2 migration issues). The Length (before hiding) of this AVP is 8.

Pseudowire Control Encapsulation (ICRQ, ICRP, ICCN, OCRQ, OCRP, OCCN)

The Pseudowire Control Encapsulation (PW Control Encapsulation) AVP, Attribute Type TBA, indicates the type of PW control encapsulation the sender of this AVP requires to be present on all incoming data packets for this L2TP session.




The Control Encapsulation Type is a 2-octet unsigned integer with the following values defined in this document:

0 - There is no control encapsulation present.
1 - The default PW control encapsulation (defined in Section 4.6) is used.

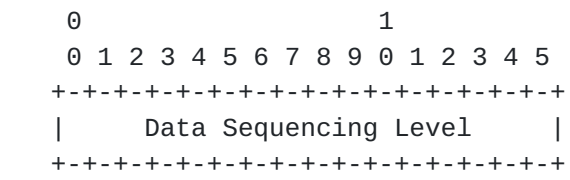
If this AVP is included in any of the above control messages and has a value other than zero, the receiving LCCE MUST include the identified control encapsulation in its outgoing data messages.

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

Data Sequencing (ICRQ, ICRP, ICCN, OCRQ, OCRP, OCCN)

The Data Sequencing AVP, Attribute Type TBA, indicates that the sender requires some or all of the incoming data packets to be sequenced.

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




The Data Sequencing Level is a 2-octet unsigned integer indicating the degree of incoming data traffic that the sender of this AVP wishes to be marked with sequence numbers.

The following values are valid data sequencing levels:

0 - No incoming data require sequencing.
1 - Only non-IP data require sequencing.
2 - All incoming data require sequencing.

If a data sequencing level of 0 is specified, there is no need to send packets with sequence numbers. If sequence numbers are sent, they will be ignored upon receipt.

If a data sequencing level of 1 is specified, only non-IP traffic carried within the given PW-specific framing should have sequence numbers applied. All traffic that can be classified as IP SHOULD be sent with no sequencing. If a packet is unable to be classified at all or if an implementation is unable to perform such classification, all packets MUST be provided with sequence numbers (essentially, a data sequencing level of 2).

If a data sequencing level of 2 is specified, all traffic MUST be sequenced.

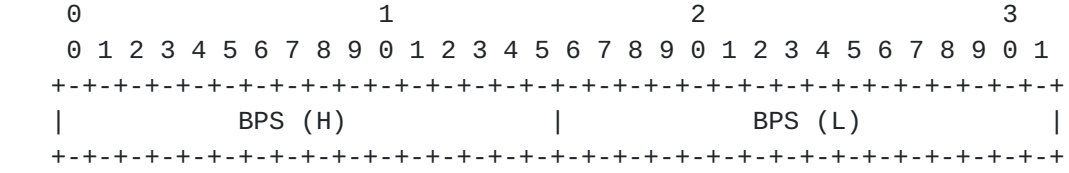
The method of sequencing is dependent upon the PW type and the PW control encapsulation. If the PW does not have any other data sequencing abilities above L2TP, a PW control encapsulation with sequence number support MUST be requested. Thus, in most cases, it is a protocol violation to send the Data Sequencing AVP without also specifying a PW control encapsulation that can be used to provide sequencing support. If such a violation occurs, the session SHOULD be disconnected with a Result Code of RC-TBA3.

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 0. The Length (before hiding) of this AVP is 6.

Tx Connect Speed (ICRQ, ICRP, ICCN)

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:




The Tx Connect Speed BPS is a 4-octet value indicating the speed in bits per second. A value of zero indicates that the speed is indeterminable or that there is no physical point-to-point link.

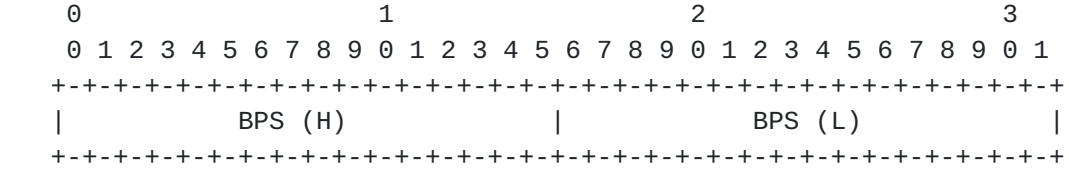
When the optional Rx Connect Speed AVP is present, the value in this AVP represents the transmit connect speed from the perspective of the LAC (e.g. data flowing from the LAC to the remote system). When the optional Rx Connect Speed AVP is NOT present, the connection speed between the remote system and 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 SHOULD be set to 1. The Length (before hiding) of this AVP is 10.

Rx Connect Speed (ICRQ, ICRP, ICCN)

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:




BPS is a 4-octet value indicating the speed in bits per second. A value of zero indicates that the speed is indeterminable or that there is no physical point-to-point link.

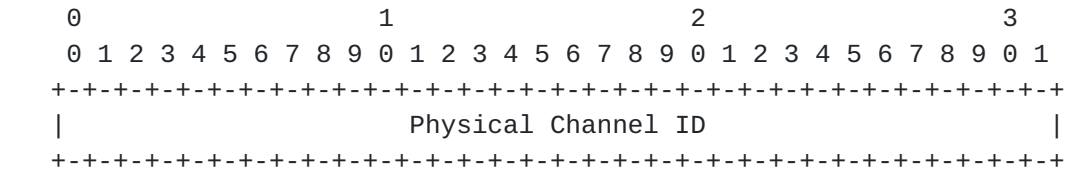
Presence of this AVP implies that the connection speed may be asymmetric with respect to the transmit connect speed given in the Tx Connect Speed AVP.

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

Physical Channel ID (ICRQ, ICRP, 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:




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

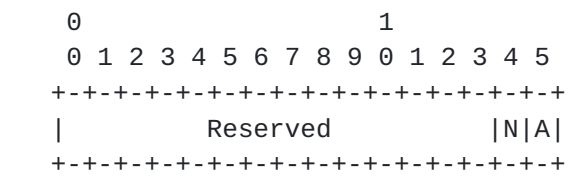
This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 0. The Length (before hiding) of this AVP is 10.

### 5.4.5 Circuit Status AVPs

Circuit Status (ICRQ, ICRP, ICCN, OCRQ, OCRP, OCCN, SLI)

The Circuit Status AVP, Attribute Type TBA, indicates the initial status of or a status change in the circuit to which the session is bound.

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




The A (Active) bit indicates whether the circuit is up/active/ready (1) or down/inactive/not-ready (0).

The N (New) bit indicates whether the circuit status indication is for a new circuit (1) or an existing circuit (0).

The remaining bits are reserved for future use. Reserved bits MUST be set to 0 when sending and ignored upon receipt.

The Circuit Status AVP is used to advertise whether a circuit or interface bound to an L2TP session is up and ready to send and/or receive traffic. Various circuit types have different names for

these status types. For instance, HDLC primary and secondary stations refer to a circuit as being "Receive Ready" or "Receive Not Ready", while Frame Relay refers to a circuit as "Active" or "Inactive". This AVP adopts the latter terminology, though the concept remains the same regardless of the PW type being tunneled.

The Circuit Status MUST be advertised in this AVP when an L2TP session is initiated by an ICRQ or OCRQ. Often, the circuit type will be marked Active when initiated, but MAY be advertised as Inactive, indicating that an L2TP session is to be created but that the interface or circuit is still not ready to pass traffic. The ICCN, OCCN, and SLI control messages all MAY contain this AVP to update the status of the circuit after establishment of the L2TP session is requested.

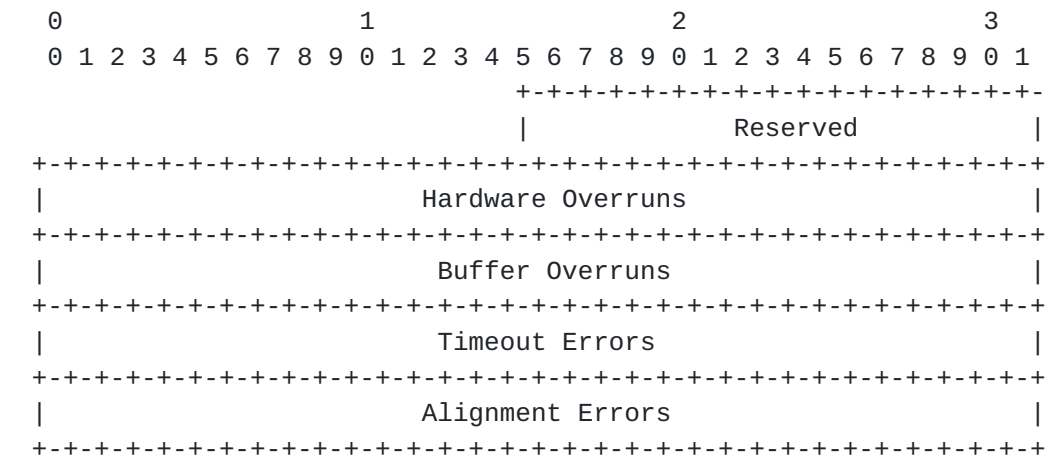
If additional circuit status information is needed for a given PW type, PW-specific AVPs MUST be defined in a separate document for that information. This AVP is only for general circuit status information applicable to all circuit/interface types.

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

Circuit Errors (WEN)

The Circuit Errors AVP, Attribute Type 34, conveys circuit error information to the peer.

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




The following fields are defined:

Reserved: 2 octets of Reserved data is present (providing longword alignment within the AVP of the following values). Reserved

data MUST be zero on sending and ignored upon receipt.
Hardware Overruns: Number of receive buffer overruns since call was established.
Buffer Overruns: Number of buffer overruns detected since call was established.
Timeout Errors: Number of timeouts 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 SHOULD be set to 1. The Length (before hiding) of this AVP is 32.

## 6. Control Connection Protocol Specification

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

The exchanges in which these messages are involved are outlined in Section 3.3.

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

Start-Control-Connection-Request (SCCRQ) is a control message used to initiate a control connection between two LCCEs. It is sent by either the LAC or the LNS to begin the control connection establishment process.

The following AVPs MUST be present in the SCCRQ:

Message Type AVP
Host Name
Assigned Control Connection ID
Pseudowire Capabilities List

The following AVPs MAY be present in the SCCRQ:

Receive Window Size
Challenge
Control Connection Tie-Breaker
Vendor Name

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

Start-Control-Connection-Reply (SCCRP) is the control message sent in reply to a received SCCRQ message. The SCCRP is used to indicate

that the SCCRQ was accepted and establishment of the control connection should continue.

The following AVPs MUST be present in the SCCRP:

- Message Type
- Host Name
- Assigned Control Connection ID
- Pseudowire Capabilities List

The following AVPs MAY be present in the SCCRP:

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

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

Start-Control-Connection-Connected (SCCCN) is the control message sent in reply to an SCCRP. The SCCCN completes the control connection 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 the control message sent by either LCCE to inform its peer that the control connection is being shut down and that the control connection should be closed. In addition, all active sessions are implicitly cleared (without sending any explicit session 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 delivery layer.

The following AVPs MUST be present in the StopCCN:

- Message Type
- Result Code

The following AVP MAY be present in the StopCCN:

Assigned Control Connection ID

Note that the Assigned Control Connection ID MUST be present if the StopCCN is sent after an SCCRQ or SCCRP message has been sent.

### 6.5 Hello (HELLO)

The Hello (HELLO) message is an L2TP control message sent by either peer of a control connection. This control message is used as a "keepalive" for the control connection. See Section 4.2 for a description of the keepalive mechanism.

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

The following AVP MUST be present in the HELLO:

Message Type

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

Incoming-Call-Request (ICRQ) is the control message sent by an LCCE to a peer when an incoming call is detected (although the ICRQ may also be sent as a result of a local event). It is the first in a three-message exchange used for establishing a session via an L2TP control connection.

The ICRQ is used to indicate that a session is to be established between an LCCE and a peer. The sender of an ICRQ provides the peer with parameter information for the session. However, the sender makes no demands about how the session is terminated at the peer (i.e. whether the L2 traffic is processed locally, forwarded, etc.).

The following AVPs MUST be present in the ICRQ:

Message Type
Local Session ID
Remote Session ID
Call Serial Number
Pseudowire Type
Pseudowire Control Encapsulation
Data Sequencing
Circuit Status

The following AVP MAY be present in the ICRQ:

Assigned Cookie
End Identifier
Session Tie-Breaker
Physical Channel ID
Tx Connect Speed
Rx Connect Speed

## 6.7 Incoming-Call-Reply (ICRP)

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

The ICRP is used to indicate that the ICRQ was successful and that the peer should establish (i.e. answer) the incoming call if it has not already done so. It also allows the sender to indicate specific parameters about the L2TP session.

The following AVPs MUST be present in the ICRP:

Message Type
Local Session ID
Remote Session ID
Pseudowire Control Encapsulation
Data Sequencing
Circuit Status

The following AVP MAY be present in the ICRP:

Assigned Cookie
Physical Channel ID
Tx Connect Speed
Rx Connect Speed

## 6.8 Incoming-Call-Connected (ICCN)

Incoming-Call-Connected (ICCN) is the control message sent by the LCCE that originally sent an ICRQ upon receiving an ICRP from its peer. It is the final message in the three-message exchange used for establishing L2TP sessions.

The ICCN is used to indicate that the ICRP was accepted, that the call has been established, and that the L2TP session should move to the established state. It also allows the sender to indicate specific parameters about the established call (parameters that may not have been available at the time the ICRQ is issued).

The following AVPs MUST be present in the ICCN:

- Message Type
- Local Session ID
- Remote Session ID

The following AVPs MAY be present in the ICCN:

- Pseudowire Control Encapsulation
- Data Sequencing
- Circuit Status
- Tx Connect Speed
- Rx Connect Speed

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

Outgoing-Call-Request (OCRQ) is the control message sent by an LCCE to an LAC to indicate that an outbound call at the LAC is to be established based on specific destination information sent in this message. It is the first in a three-message exchange used for establishing a session and placing a call on behalf of the initiating LCCE.

Note that a call may be any L2 connection requiring well-known destination information to be sent from an LCCE to an LAC. This call could be a dialup connection to the PSTN, an SVC connection, the IP address of another LCCE, or any other destination dictated by the sender of this message.

The following AVPs MUST be present in the OCRQ:

- Message Type
- Local Session ID
- Remote Session ID
- Call Serial Number
- Pseudowire Type
- Pseudowire Control Encapsulation
- Data Sequencing
- Circuit Status

The following AVPs MAY be present in the OCRQ:

- Assigned Cookie
- End Identifier
- Session Tie-Breaker

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

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

OCRP is used to indicate that the LAC has been able to attempt the outbound call. The message returns any relevant parameters regarding the call attempt. Data MUST not be forwarded until the OCCN is received indicating that the call has been placed.

The following AVPs MUST be present in the OCRP:

- Message Type
- Local Session ID
- Remote Session ID
- Pseudowire Control Encapsulation
- Data Sequencing
- Circuit Status

The following AVPs MAY be present in the OCRP:

- Assigned Cookie
- Physical Channel ID

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

Outgoing-Call-Connected (OCCN) is the control message sent by an LAC to another LCCE after 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.

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

The following AVPs MUST be present in the OCCN:

- Message Type
- Local Session ID
- Remote Session ID

The following AVPs MAY be present in the OCCN:

- Pseudowire Control Encapsulation
- Data Sequencing

Circuit Status

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

The Call-Disconnect-Notify (CDN) is a control message sent by an LCCE to request disconnection of a specific session. Its purpose is to inform the peer of the disconnection and the reason for the disconnection. 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
Local Session ID
Remote Session ID
Result Code

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

The WAN-Error-Notify (WEN) is a control message sent from an LAC to an LNS to indicate WAN error conditions. 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
Circuit Errors
Local Session ID
Remote Session ID

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

The Set-Link-Info control message is sent by an LCCE to convey link or circuit status change information regarding the circuit associated with this L2TP session. For example, if PPP renegotiates LCP or a Frame Relay VC transitions to Active or Inactive, an SLI message SHOULD be sent to indicate this event. Precise details of when the SLI is sent, what PW type-specific AVPs must be present, and how those AVPs should be interpreted by the receiving peer are outside the scope of this document. These details should be described in the associated payload-specific documents that require use of this message.

The following AVPs MUST be present in the SLI:

Message Type

Local Session ID
Remote Session ID

The following AVPs MAY be present in the SLI:

Circuit Status

## 7. Control Connection State Machines

The state tables defined in this section govern the exchange of control messages defined in Section 6. Tables are defined for incoming call placement and outgoing call placement, as well as for initiation of the control connection itself. The state tables do not encode timeout and retransmission behavior, as this is handled in the underlying reliable control message delivery mechanism (see Section 4.2).

### 7.1 Malformed Control Messages

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 (1) a message that contains a Message Type marked as mandatory (see Section 5.4.1) but that is unknown to the implementation, or (2) a control message that is received in the wrong state.

Examples of malformed control messages include (1) a message that has an invalid value in its header, (2) a message that contains an AVP that is formatted incorrectly or whose value is out of range, and (3) a message that is missing a required AVP. A control message with a malformed header MUST be discarded.

If a malformed AVP is received with the M bit set, the session or control connection MUST be terminated with a proper Result or Error Code sent. A malformed yet non-mandatory (M bit is not set) AVP within a control message should be handled like an unrecognized non-mandatory AVP. That is, the AVP MUST be ignored (with the exception of logging a local error message), and the message MUST be accepted.

This policy MUST NOT be considered a license to send malformed AVPs, but rather, 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 control connection destruction, the reliable delivery mechanism must be allowed to run (see Section 4.2) before destroying the control connection. This permits the control connection management messages to be reliably delivered to the peer.

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

## 7.2 Timing Considerations

Due to the real-time nature of L2 circuit signaling, an LCCE should be implemented using a multi-threaded architecture 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.

## 7.3 Control Connection States

The L2TP control connection protocol is not distinguishable between the two LCCEs but is distinguishable between the originator and receiver. The originating peer is the one that first initiates establishment of the control connection. (In a tie-breaker situation, this is the winner of the tie.) Since either the LAC or the LNS can be the originator, a collision can occur. See the Control Connection Tie-Breaker AVP in Section 5.4.3 for a description of this and its resolution.

| 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 control-conn 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 control-conn 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 control-conn open event to waiting sessions | established |
| established | Administrative control-conn close event | Send StopCCN, clean up | idle |
| established | Receive SCCRQ, SCCRP, SCCCN | Send StopCCN, clean up | idle |
| idle, wait-ctl-reply, wait-ctl-conn, established | Receive StopCCN | Clean up | idle |

The states associated with an LCCE for control connection establishment are as follows:

idle
   Both initiator and recipient start from this state. An initiator

transmits an SCCRQ, while a recipient remains in the idle state until receiving an SCCRQ.

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 5.4.3.

wait-ctl-conn
Awaiting an SCCCN. Upon receipt, the challenge response contained in the message is checked. The control connection is established if authentication succeeds; otherwise, it is torn down.

established
An established connection may be terminated by either a local condition or the receipt of a StopCCN. In the event of a local termination, the originator MUST send a StopCCN and clean up the control connection. If the originator receives a StopCCN, it MUST also clean up the control connection.

## 7.4 Incoming Calls

An ICRQ is generated by an LCCE, typically in response to an incoming call or a local event. Once the LCCE sends the ICRQ, it waits for a response from the peer. However, it may choose to postpone establishment of the call (e.g. answering the call, bringing up the circuit) until the peer has indicated with an ICRP that it will accept the call. The peer may choose not to accept the call if, for instance, there are insufficient resources to handle an additional session.

If the peer chooses to accept the call, it responds with an ICRP. When the local LCCE receives the ICRP, it attempts to establish the call. A final call connected message, the ICCN, is sent from the local LCCE to the peer to indicate that the call states for both LCCEs should enter the established state. If the call is terminated before the peer can accept it, a CDN is sent by the local LCCE to indicate this condition.

When a call transitions to a "disconnected" or "down" state, the call is cleared normally, and the local LCCE sends a CDN. Similarly, if the peer wishes to clear a call, it sends a CDN and cleans up its session.

### 7.4.1 ICRQ Sender States

| State | Event | Action | New State |
|---|---|---|---|
| idle | Call signal or ready to receive incoming conn | Initiate local control-conn open | wait-control-conn |
| idle | Receive ICCN, ICRP, CDN | Clean up | idle |
| wait-control-conn | Bearer line drop or local close request | Clean up | idle |
| wait-control-conn | control-conn-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 | Send CDN, clean up | idle |
| established | Receive CDN | Clean up | idle |
| established | Receive ICRQ, ICRP, ICCN | Send CDN, clean up | idle |
| established | Local close request | Send CDN, clean up | idle |

The states associated with the ICRQ sender are as follows:

idle
The LCCE detects an incoming call on one of its interfaces (e.g. an analog PSTN line rings, or an ATM PVC is provisioned), or a local event occurs. The LCCE initiates its control connection establishment state machine and moves to a state waiting for

confirmation of the existence of a control connection.

wait-control-connection
In this state, the session is waiting for either the control connection to be opened or for verification that the control connection is already open. Once an indication that the control connection has been opened is received, session control messages may be exchanged. The first of these messages is the ICRQ.

wait-reply
The ICRQ sender receives either (1) a CDN indicating the peer is not willing to accept the call (general error or do not accept) and moves back into the idle state, or (2) an ICRP indicating the call is accepted. In the latter case, the LCCE sends an ICCN and enters the established state.

established
Data is exchanged over the session. The call may be cleared by any of the following:
+ An event on the connected interface: The LCCE sends a CDN.
+ Receipt of a CDN: The LCCE cleans up, disconnecting the call.
+ A local reason: The LCCE sends a CDN.

### 7.4.2 ICRQ Recipient 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 | Local close request | Send CDN, clean up | idle |
| established | Receive ICRQ, ICRP, ICCN | Send CDN, clean up | idle |

The states associated with the ICRQ recipient are as follows:

idle
: An ICRQ is received. If the request is not acceptable, a CDN is sent back to the peer LCCE, and the local LCCE remains in the idle state. If the ICRQ is acceptable, an ICRP is sent. The session moves to the wait-connect state.

wait-connect
: The local LCCE is waiting for an ICCN from the peer. Upon receipt of the ICCN, the local LCCE moves to established state.

established
: The session is terminated either by sending a CDN or by receiving a CDN from the peer. Clean up follows on both sides regardless of the initiator.

## 7.5 Outgoing Calls

Outgoing calls instruct an LAC to place a call. There are three messages for outgoing calls: OCRQ, OCRP, and OCCN. An LCCE first sends an OCRQ to an LAC to request an outgoing call. The LAC MUST respond to the OCRQ with an OCRP once it determines that the proper facilities exist to place the call and that the call is administratively authorized. Once the outbound call is connected, the LAC sends an OCCN to the peer indicating the final result of the call attempt.

### 7.5.1 OCRQ Sender States

| State | Event | Action | New State |
|---|---|---|---|
| idle | Local open request | Initiate local control-conn-open | wait-control-conn |
| idle | Receive OCCN, OCRP | Clean up | idle |

| | | | |
|---|---|---|---|
| wait-control-conn | control-conn-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-control-conn | Local close request | Clean up | idle |

The states associated with the OCRQ sender are as follows:

idle, wait-control-conn
: When an outgoing call request is initiated, a control connection is created as described above, if not already present. Once the control connection is established, an OCRQ is sent to the LAC, and the session moves into the wait-reply state.

wait-reply
: If a CDN is received, the session is cleaned up and returns to idle state. If an OCRP is received, the call is in progress, and the session moves to the wait-connect state.

wait-connect
: If a CDN is received, the session is cleaned up and returns to idle state. If an OCCN is received, the call has succeeded, and

the session may now exchange data.

established
If a CDN is received, the session is cleaned up and returns to idle state. Alternatively, if the LCCE chooses to terminate the session, it sends a CDN to the LAC, cleans up the session, and moves the session to idle state.

### 7.5.2 OCRQ Recipient (LAC) States

| State | Event | Action | New State |
|---|---|---|---|
| idle | Receive OCRQ, acceptable | Send OCRP, Place call | 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 | Call placement successful | Send OCCN | established |
| wait-cs-answer | Call placement failed | 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 | Local close request | Send CDN, clean up | idle |

The states associated with the LAC for outgoing calls are as follows:

idle
If the OCRQ is received in error, respond with a CDN. Otherwise, place the call, send an OCRP, and move to the wait-cs-answer state.

wait-cs-answer
If the call is not completed or a timer expires while waiting for the call to complete, send a CDN with the appropriate error condition set, and go to idle state. If a circuit-switched connection is established, send an OCCN indicating success, and go to established state.

established
If the LAC receives a CDN from the peer, the call MUST be released via appropriate mechanisms, and the session cleaned up. If the call is disconnected because the circuit transitions to a "disconnected" or "down" state, the LAC MUST send a CDN to the peer and return to idle state.

### 7.6 Termination of a Control Connection

The termination of a control connection consists of either peer issuing a StopCCN. The sender of this message SHOULD wait a finite period of time for the acknowledgment of this message before releasing the control information associated with the control connection. The recipient of this message should send an acknowledgment of the message to the peer, then release the associated control information.

When to release a control connection 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 control connection open for a period of time or perhaps indefinitely after the last session for that control connection is cleared. Others may choose to disconnect the control connection immediately after the last call on the control connection disconnects.

## 8. Security Considerations

This section addresses some of the security issues that L2TP encounters in its operation.

### 8.1 Control Connection Endpoint Security

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

establishment has been completed successfully.

For authentication to occur, the LCCE pair 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 control connection authentication AVPs include differentiating values in the CHAP ID fields for each message digest calculation to guard against replay attacks.

The Assigned Control Connection ID and Assigned Session ID (see Section 5.4) 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 LCCEs.

The Assigned Cookie value MUST be selected in an unpredictable manner. However, the Cookie MUST not be regarded as packet-level authentication or security of any kind. It should be used for nothing more than simple configuration error detection and identification of misrouted packets. Since the Cookie is sent and advertised in the clear, it is by no means a true packet-level security measure, such as that offered by IPsec.

## 8.2 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 the data being carried on an L2TP data session. As such, L2TP is only concerned with confidentiality, authenticity, and integrity of the L2TP packets between two LCCEs, not unlike link layer encryption being concerned only about protecting the confidentiality of traffic between the physical endpoints.

## 8.3 End-to-End Security

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

## 8.4 L2TP and IPsec

When running over IP, IPsec provides packet-level security via ESP [RFC3193]. All L2TP control and data packets for a particular control connection 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 network layer above L2TP. These network layer access control features may be handled at an LCCE via vendor-specific authorization features based upon the authenticated 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.

### 8.5 Impact of L2TPv3 Features on RFC 3193

[RFC3193] defines the recommended method for securing L2TP as defined in [RFC2661]. L2TP as defined in this document should possess the same interface to IPsec as [RFC2661] when running on UDP/IP. UDP has the added advantage of being able to provide a native method for IPsec to distinguish multiple Security Associations (presumably with different policies) between the same control connection endpoints without having to extend the definitions of IPsec or allocate additional IP addresses between endpoints. Therefore, when securing L2TP with IPsec via [RFC3193], L2TPv3 MUST operate over UDP/IP as described in Section 4.1.2.

## 9. 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.

### 9.1 AVP Attributes

As defined in Section 5.1, AVPs contain Vendor ID, Attribute, and Value fields. For a Vendor ID value of 0, IANA will maintain a registry of assigned Attributes and, in some cases, Values. Attributes 0-39 are assigned as defined in Section 5.4. The remaining values are available for assignment upon Expert Review

[RFC2434].

## 9.2 Message Type AVP Values

As defined in Section 5.4.1, Message Type AVPs (Attribute Type 0) have an associated value maintained by IANA. Values 0-16 are defined in Section 3.1. The remaining values are available for assignment upon Expert Review [RFC2434].

## 9.3 Result Code AVP Values

As defined in Section 5.4.2, 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.

### 9.3.1 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 5.4.2; for the CDN 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 upon Expert Review [RFC2434].

### 9.3.2 Error Code Field Values

Values 0-9 are defined in Section 5.4.2. The remaining values are available for assignment upon Expert Review [RFC2434].

## 9.4 AVP Header Bits

There are four remaining reserved bits in the AVP header. Additional bits should only be assigned via a Standards Action [RFC2434].

## 9.5 L2TP Control Message Header Bits

There are nine remaining reserved bits in the control message header. Additional bits should only be assigned via a Standards Action [RFC2434].

Care should be taken before using reserved bits 6 and 7 in the L2TPv3 control message header since these bits have meaning for L2TPv2 data messages. Using these two bits in L2TPv3 MAY trigger an unforeseen interoperability problem with L2TPv3 implementations based on L2TPv2. Therefore, it is recommended that these two bits be utilized last, after the other reserved bits have been assigned roles.

## 10. 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 Postel, J., "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 Coradetti, T., "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 Lear, E., "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 Willens, S., "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 Kolar, T., "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 Zorn, G., "Point-to-Point Tunneling Protocol (PPTP)", RFC 2637, July 1999.

[RFC2661] Townsley, W., et al., "Layer Two Tunneling Layer Two Tunneling Protocol (L2TP)", RFC 2661, August 1999.

[RFC2764] Gleeson, B., Lin, A., Heinanen, J., Finland, T., Armitage, G., and Malis, A., "A Framework for IP Based Virtual Private Networks", RFC 2764, February 2000.

[RFC2809] Aboba, B., and Zorn, G., "Implementation of L2TP Compulsory Tunneling via RADIUS", RFC 2809, April 2000.

[RFC3193] Patel, B., Aboba, B., Dixon, W., Zorn, G., and Booth, S., "Securing L2TP using IPsec", RFC 3193, November 2001.

[RFC3070] Rawat, V., Tio, R., Nanji, S., and Verma, R., "Layer Two Tunneling Protocol (L2TP) over Frame Relay", RFC 3070, February 2001.

[STEVENS] Stevens, W. Richard, "TCP/IP Illustrated, Volume I: The Protocols", Addison-Wesley Publishing Company, Inc., March 1996, ISBN 0-201-63346-9.

[L2TPAAL5] Davison, M., Lin, A., Singh, A., Stephens, J., Turner, R., Tio, R., and Nanji, S., "L2TP Over AAL5," Internet Draft, August 2001, draft-ietf-l2tpext-l2tp-atm-02.txt.

## 11. Editors' Addresses

Jed Lau
cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
jedlau@cisco.com

Gurdeep Singh Pall
Microsoft Corporation
Redmond, WA
gurdeep@microsoft.com

Bill Palter
RedBack Networks, Inc
1389 Moffett Park Drive
Sunnyvale, CA 94089
palter@zev.net

Allan Rubens
acr@del.com

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

Andrew J. Valencia
P.O. Box 2928
Vashon, WA 98070
vandys@zendo.com

Glen Zorn
cisco Systems
500 108th Avenue N.E., Suite 500
Bellevue, WA 98004
gwz@cisco.com

## Appendix A: Control 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 acknowledgment (either explicit or piggybacked). When the acknowledgment 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/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 Control Connection Establishment

In this example, an LCCE establishes a control connection, 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 acknowledgment within a message sent as a reply to the ICRQ or OCRQ that will likely follow from the side that initiated the control connection.

```
          LCCE A                  LCCE B
          ------                  ------
          SCCRQ     ->
          Nr: 0, Ns: 0

                                  <-     SCCRP
                                  Nr: 1, Ns: 0
          SCCCN     ->
          Nr: 1, Ns: 1

                                  <-       ZLB
                                  Nr: 2, Ns: 1
```

B.2: Lost Packet with Retransmission

An existing control connection has a new session requested by LCCE A. The ICRP is lost and must be retransmitted by LCCE B. Note that loss of the ICRP has two effects: It not only keeps the upper level state machine from progressing, but also keeps LCCE A from seeing a timely lower level acknowledgment of its ICRQ.

```
          LCCE A                          LCCE B
          ------                          ------
          ICRQ      ->
          Nr: 1, Ns: 2
                          (packet lost)   <-      ICRP
                                          Nr: 3, Ns: 1

        (pause; LCCE A's timer started first, so fires first)

          ICRQ      ->
          Nr: 1, Ns: 2

        (Realizing that it has already seen this packet,
         LCCE B discards the packet and sends a ZLB)

                                          <-       ZLB
                                          Nr: 3, Ns: 2

        (LCCE B's retransmit timer fires)

                                          <-      ICRP
                                          Nr: 3, Ns: 1
          ICCN      ->
          Nr: 2, Ns: 3

                                          <-       ZLB
                                          Nr: 4, Ns: 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 that 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.

## Appendix D: Full Copyright Statement