Network Working Group
Request for Comments: 3931
Category: Standards Track

J. Lau, Ed.
M. Townsley, Ed.
Cisco Systems
I. Goyret, Ed.
Lucent Technologies

March 2005

Layer Two Tunneling Protocol - Version 3 (L2TPv3)

Status of this Memo

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

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

This document describes "version 3" of the Layer Two Tunneling Protocol (L2TPv3). L2TPv3 defines the base control protocol and encapsulation for tunneling multiple Layer 2 connections between two IP nodes. Additional documents detail the specifics for each data link type being emulated.

Table of Contents

1. Introduction 3
   1.1. Changes from RFC 2661. 4
   1.2. Specification of Requirements. 4
   1.3. Terminology. 5
2. Topology 8
3. Protocol Overview. 9
   3.1. Control Message Types. 10
   3.2. L2TP Header Formats. 11
      3.2.1. L2TP Control Message Header. 11
      3.2.2. L2TP Data Message. 12
   3.3. Control Connection Management. 13
      3.3.1. Control Connection Establishment 14
      3.3.2. Control Connection Teardown. 14
   3.4. Session Management 15
      3.4.1. Session Establishment for an Incoming Call . . . 15
      3.4.2. Session Establishment for an Outgoing Call . . . 15

3.4.3. Session Teardown 16
4. Protocol Operation 16
4.1. L2TP Over Specific Packet-Switched Networks (PSNs) . . . 16
4.1.1. L2TPv3 over IP 17
4.1.2. L2TP over UDP. 18
4.1.3. L2TP and IPsec 20
4.1.4. IP Fragmentation Issues. 21
4.2. Reliable Delivery of Control Messages. 23
4.3. Control Message Authentication 25
4.4. Keepalive (Hello). 26
4.5. Forwarding Session Data Frames 26
4.6. Default L2-Specific Sublayer 27
4.6.1. Sequencing Data Packets. 28
4.7. L2TPv2/v3 Interoperability and Migration 28
4.7.1. L2TPv3 over IP 29
4.7.2. L2TPv3 over UDP. 29
4.7.3. Automatic L2TPv2 Fallback. 29
5. Control Message Attribute Value Pairs. 30
5.1. AVP Format 30
5.2. Mandatory AVPs and Setting the M Bit 32
5.3. Hiding of AVP Attribute Values 33
5.4. AVP Summary. 36
5.4.1. General Control Message AVPs 36
5.4.2. Result and Error Codes 40
5.4.3. Control Connection Management AVPs 43
5.4.4. Session Management AVPs. 48
5.4.5. Circuit Status AVPs. 57
6. Control Connection Protocol Specification. 59
6.1. Start-Control-Connection-Request (SCCRQ) 60
6.2. Start-Control-Connection-Reply (SCCRP) 60
6.3. Start-Control-Connection-Connected (SCCCN) 61
6.4. Stop-Control-Connection-Notification (StopCCN) 61
6.5. Hello (HELLO). 61
6.6. Incoming-Call-Request (ICRQ) 62
6.7. Incoming-Call-Reply (ICRP) 63
6.8. Incoming-Call-Connected (ICCN) 63
6.9. Outgoing-Call-Request (OCRQ) 64
6.10. Outgoing-Call-Reply (OCRP) 65
6.11. Outgoing-Call-Connected (OCCN) 65
6.12. Call-Disconnect-Notify (CDN) 66
6.13. WAN-Error-Notify (WEN) 66
6.14. Set-Link-Info (SLI). 67
6.15. Explicit-Acknowledgement (ACK) 67
7. Control Connection State Machines. 68
7.1. Malformed AVPs and Control Messages. 68
7.2. Control Connection States. 69
7.3. Incoming Calls 71
7.3.1. ICRQ Sender States 72

7.3.2. ICRQ Recipient States. 73
7.4. Outgoing Calls 74
7.4.1. OCRQ Sender States 75
7.4.2. OCRQ Recipient (LAC) States. 76
7.5. Termination of a Control Connection. 77
8. Security Considerations. 78
8.1. Control Connection Endpoint and Message Security 78
8.2. Data Packet Spoofing 78
9. Internationalization Considerations. 79
10. IANA Considerations. 80
10.1. Control Message Attribute Value Pairs (AVPs) 80
10.2. Message Type AVP Values. 81
10.3. Result Code AVP Values 81
10.4. AVP Header Bits. 82
10.5. L2TP Control Message Header Bits 82
10.6. Pseudowire Types 83
10.7. Circuit Status Bits. 83
10.8. Default L2-Specific Sublayer bits. 84
10.9. L2-Specific Sublayer Type. 84
10.10 Data Sequencing Level. 84
11. References 85
11.1. Normative References 85
11.2. Informative References 85
12. Acknowledgments. 87
Appendix A: Control Slow Start and Congestion Avoidance. 89
Appendix B: Control Message Examples 90
Appendix C: Processing Sequence Numbers. 91
Editors' Addresses 93
Full Copyright Statement 94

1. Introduction

The Layer Two Tunneling Protocol (L2TP) provides a dynamic mechanism for tunneling Layer 2 (L2) "circuits" across a packet-oriented data network (e.g., over IP). L2TP, as originally defined in RFC 2661, is a standard method for tunneling Point-to-Point Protocol (PPP) [RFC1661] 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 defined in this document 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 nodes across an IP network. Additional documents are expected to be published for each L2 data link emulation type (a.k.a. pseudowire-type) supported by L2TP (i.e., PPP, Ethernet, Frame Relay, etc.). These documents will

contain any pseudowire-type specific details that are outside the scope of this base specification.

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 header. (Layer 2 Forwarding, L2F, [RFC2341] was 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.

1.1. Changes from RFC 2661

Many 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 reuse, interoperability experience, and a directed evolution of L2TP as it is applied to new tasks.

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.

Extension of the Tunnel Authentication mechanism to cover the entire control message rather than just a portion of certain messages.

Details of these changes and a recommendation for transitioning to L2TPv3 are discussed 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), a length field, and a Value containing the actual value identified by the attribute. Zero or more AVPs make up the body of control messages, which are used in the establishment, maintenance, and teardown of control connections. This basic construct is sometimes referred to as a Type-Length-Value (TLV) in some specifications. (See also: Control Connection, Control Message.)

Call (Circuit Up)

The action of transitioning a circuit on an L2TP Access Concentrator (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 Public Switched Telephone Network (PSTN)) or statically configured (e.g., provisioning a Virtual Circuit 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.)

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, an IEEE 802 VLAN, 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 signaling to govern the establishment, maintenance, and teardown of the circuit. For the purposes of this document, a statically configured circuit is considered to be essentially the same as a very simple, long-lived, 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 connection 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. (a.k.a. Data Packet, See also: Data Channel.)

Data Channel

The channel for L2TP-encapsulated data traffic that passes between two LCCEs over a Packet-Switched Network (i.e., IP). (See also: Control Connection, Data Message.)

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)

If an L2TP Control Connection Endpoint (LCCE) is being used to cross-connect an L2TP session directly to a data link, we refer to it as an L2TP Access Concentrator (LAC). An LCCE may act as both an L2TP Network Server (LNS) for some sessions and an LAC for others, so these terms must only be used within the context of a given set of sessions unless the LCCE is in fact single purpose for a given topology. (See also: LCCE, LNS.)

L2TP Control Connection Endpoint (LCCE)

An L2TP node that exists at either end of an L2TP control connection. May also be referred to as an LAC or LNS, depending on whether tunneled frames are processed at the data link (LAC) or network layer (LNS). (See also: LAC, LNS.)

L2TP Network Server (LNS)

If a given L2TP session is terminated at the L2TP node and the encapsulated network layer (L3) packet processed on a virtual interface, we refer to this L2TP node as an L2TP Network Server

(LNS). A given LCCE may act as both an LNS for some sessions and an LAC for others, so these terms must only be used within the context of a given set of sessions unless the LCCE is in fact single purpose for a given topology. (See also: LCCE, LAC.)

Outgoing Call

The action of placing a call by an LAC, typically in response to policy directed by the peer in an Outgoing Call Request. (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 not known a priori by the LAC (e.g., a number to dial). (See also: Call, Incoming Call, Outgoing Call.)

Packet-Switched Network (PSN)

A network that uses packet switching technology for data delivery. For L2TPv3, this layer is principally IP. Other examples include MPLS, Frame Relay, and ATM.

Peer

When used in context with L2TP, Peer refers to the far end of an L2TP control connection (i.e., the remote 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 the entity that is created between two LCCEs in order to exchange parameters for and maintain an emulated L2 connection. Multiple sessions may be associated with a single Control Connection.

Zero-Length Body (ZLB) Message

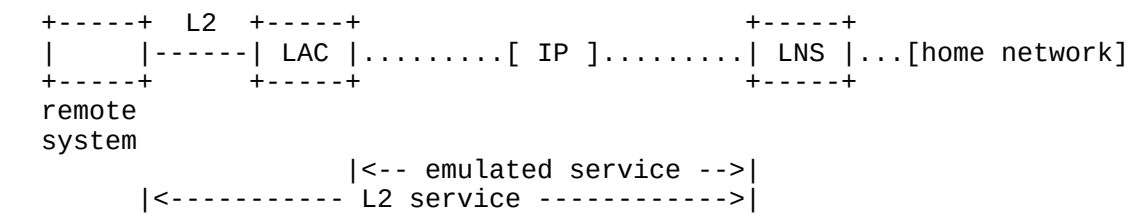
A control message with only an L2TP header. ZLB messages are used only to acknowledge messages on the L2TP reliable control connection. (See also: Control Message.)

2. Topology

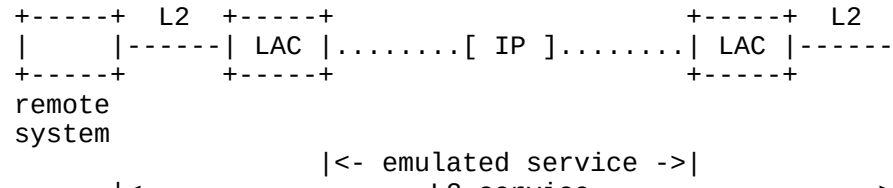
L2TP operates between two L2TP Control Connection Endpoints (LCCEs), tunneling traffic across a packet network. There are three predominant tunneling models in which L2TP operates: LAC-LNS (or vice versa), LAC-LAC, and LNS-LNS. These models are diagrammed below. (Dotted lines designate network connections. Solid lines designate circuit connections.)

Figure 2.0: L2TP Reference Models

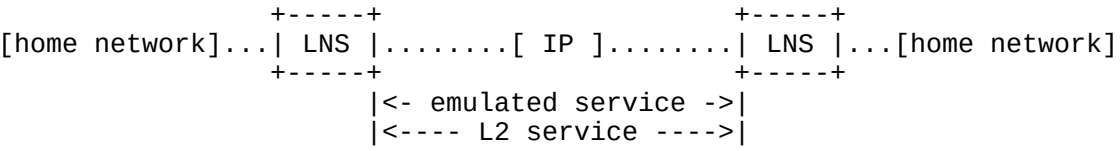
(a) LAC-LNS Reference Model: On one side, the LAC receives traffic from an L2 circuit, which it forwards via L2TP across an IP or other packet-based network. On the other side, an LNS logically terminates the L2 circuit locally and routes network traffic to the home network. The action of session establishment is driven by the LAC (as an incoming call) or the LNS (as an outgoing call).



(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. In its simplest form, an LAC acts as a simple cross-connect between a circuit to a remote system 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 simultaneously, in which a tie breaking mechanism is utilized).



(c) LNS-LNS Reference Model: This model has two LNSs as the LCCEs. A user-level, traffic-generated, or signaled event typically drives session establishment from one side of the tunnel. For example, a tunnel generated from a PC by a user, or automatically by customer premises equipment.



Note: In L2TPv2, user-driven tunneling of this type is often referred to as "voluntary tunneling" [RFC2809]. Further, an LNS acting as part of a software package on a host is sometimes referred to as an "LAC Client" [RFC2661].

3. Protocol Overview

L2TP is comprised of two types of messages, control messages and data messages (sometimes referred to as "control packets" and "data packets", respectively). 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.

The L2TPv3 control message format defined in this document borrows largely from L2TPv2. These control messages are used in conjunction with the associated protocol state machines that govern the dynamic setup, maintenance, and teardown for L2TP sessions. The data message format for tunneling data packets may be utilized with or without the L2TP control channel, either via manual configuration or via other signaling methods to pre-configure or distribute L2TP session information. Utilization of the L2TP data message format with other signaling methods is outside the scope of this document.

Figure 3.0: L2TPv3 Structure

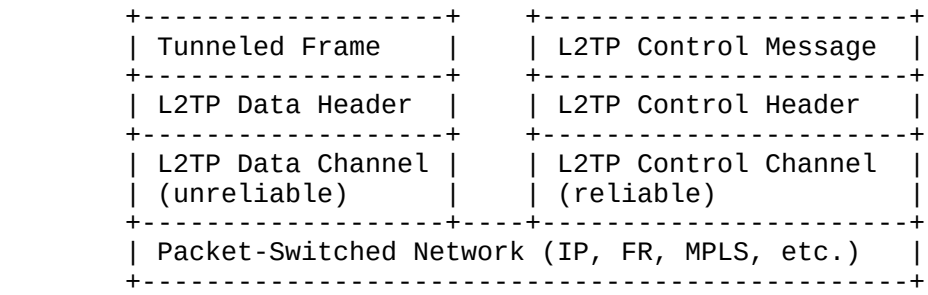


Figure 3.0 depicts the relationship of control messages and data messages over the L2TP control and data channels, respectively. Data messages are passed over an unreliable data channel, encapsulated by an L2TP header, and sent over a Packet-Switched Network (PSN) such as IP, UDP, Frame Relay, ATM, MPLS, 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, 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 6.15 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
20	(ACK)	Explicit Acknowledgement

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

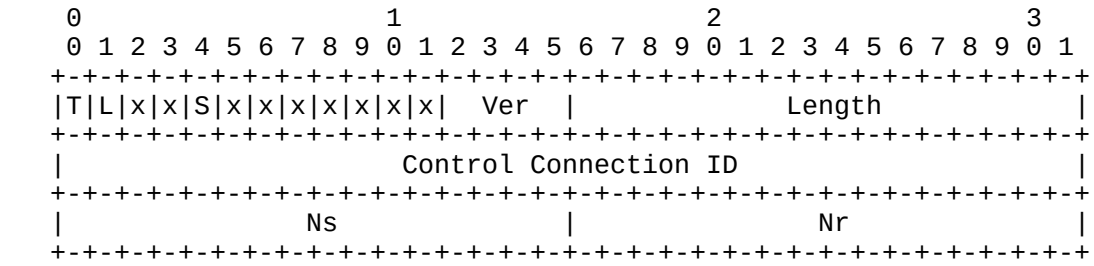
This section defines header formats for L2TP control messages and L2TP data messages. All values are placed into their respective fields and sent in network order (high-order octets first).

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.

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 that includes L2TPv2 [RFC2661] and/or L2F [RFC2341] as well, 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.

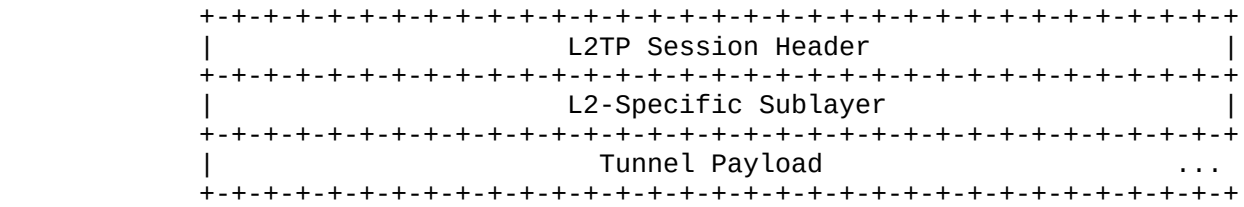
Ns indicates the sequence number for this control message, beginning at zero and incrementing by one (modulo 2**16) for each message sent. See Section 4.2 for more information on using this field.

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 L2-Specific Sublayer, and (3) the Tunnel Payload, as depicted below.

Figure 3.2.2: L2TP Data Message Header



The L2TP Session Header is specific to the encapsulating PSN over which the L2TP traffic is delivered. The Session Header MUST provide (1) a method of distinguishing traffic among multiple L2TP data sessions and (2) a method of distinguishing data messages from control messages.

Each type of encapsulating 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 L2-Specific Sublayer is an intermediary layer between the L2TP session header and the start of the tunneled frame. It contains control fields that are used to facilitate the tunneling of each frame (e.g., sequence numbers or flags). The Default L2-Specific Sublayer for L2TPv3 is defined in Section 4.6.

The Data Message Header is followed by the Tunnel Payload, 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 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, Explicit Acknowledgement (ACK) messages 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.

LCCEs are identified during control connection establishment either by the Host Name AVP, the Router ID AVP, or a combination of the two (see Section 5.4.3). The identity of a peer LCCE is central to selecting proper configuration parameters (i.e., Hello interval, window size, etc.) for a control connection, as well as for determining how to set up associated sessions within the control connection, password lookup for control connection authentication, control connection level tie breaking, etc.

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 an ACK message 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 ACK message 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
(Call
 Accepted)

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 ->

(Call Operation
 Completed
 Successfully)
```

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

4.1. L2TP Over Specific Packet-Switched Networks (PSNs)

L2TP may operate over a variety of PSNs. There are two modes described for operation over IP, L2TP directly over IP (see Section 4.1.1) and L2TP over UDP (see Section 4.1.2). L2TPv3 implementations MUST support L2TP over IP and SHOULD support L2TP over UDP for better NAT and firewall traversal, and for 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 [RFC3355].

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. The Session ID alone provides the necessary context for all further packet processing, including the presence, size, and value of the Cookie, the type of L2-Specific Sublayer, and the type of payload being tunneled.

Cookie

The optional Cookie field contains a variable-length value (maximum 64 bits) used to check the association of a received data message with the session identified by the Session ID. The Cookie MUST be set to the configured or signaled random value for this session. The Cookie provides an additional level of guarantee that a data message has been directed to the proper session by the Session ID. A well-chosen Cookie may prevent inadvertent misdirection of stray packets with recently reused Session IDs, Session IDs subject to packet corruption, etc. The Cookie may also provide protection against some specific malicious packet insertion attacks, as described in Section 8.2.

When the L2TP control connection is used for session establishment, random Cookie values are selected and exchanged as Assigned Cookie AVPs during session creation.

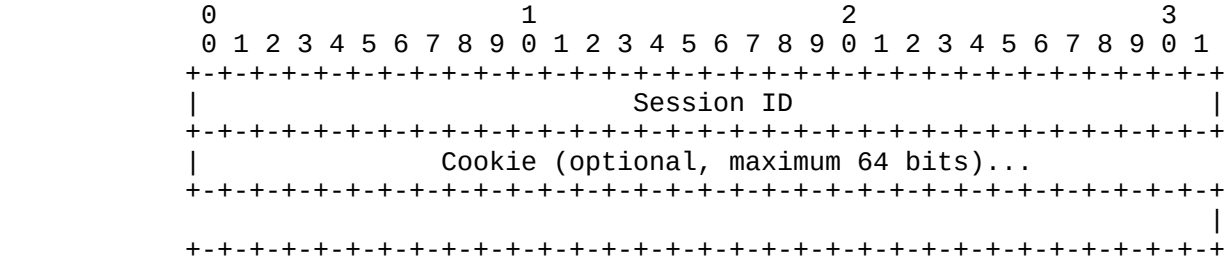
4.1.1. L2TPv3 over IP

L2TPv3 over IP (both versions) utilizes the IANA-assigned IP protocol ID 115.

4.1.1.1. L2TPv3 Session Header Over IP

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 designed to optimize packet processing. The following session header format is utilized when operating L2TPv3 over IP:

Figure 4.1.1.1: L2TPv3 Session Header Over IP



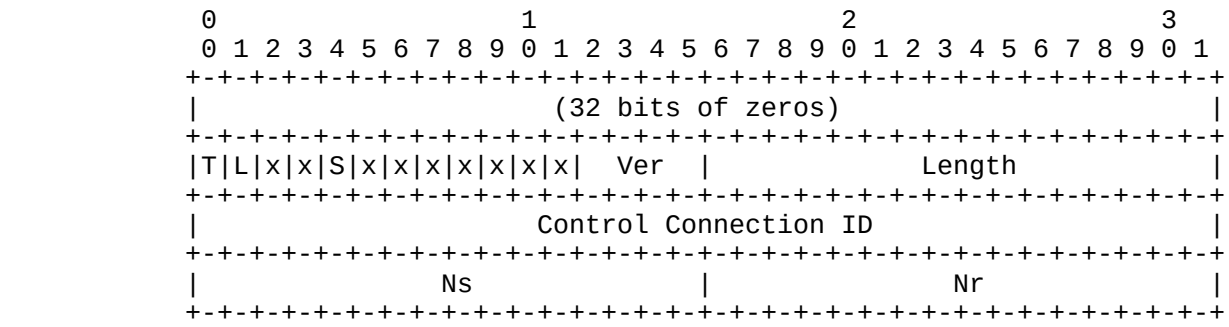
The Session ID and Cookie fields are as defined in Section 4.1. The Session ID of zero is reserved for use by L2TP control messages (see Section 4.1.1.2).

4.1.1.2. L2TP Control and Data Traffic over IP

Unlike L2TP over UDP, which uses the T bit to distinguish between L2TP control and data packets, L2TP over IP uses the reserved Session ID of zero (0) when sending control messages. It is presumed that checking for the zero Session ID is more efficient -- both in header size for data packets and in processing speed for distinguishing between control and data messages -- than checking a single bit.

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

Figure 4.1.1.2: L2TPv3 Control Message Header Over IP



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.

When operating directly over IP, L2TP packets lose the ability to take advantage of the UDP checksum as a simple packet integrity check, which is of particular concern for L2TP control messages. Control Message Authentication (see Section 4.3), even with an empty password field, provides for a sufficient packet integrity check and SHOULD always be enabled.

4.1.2. L2TP over UDP

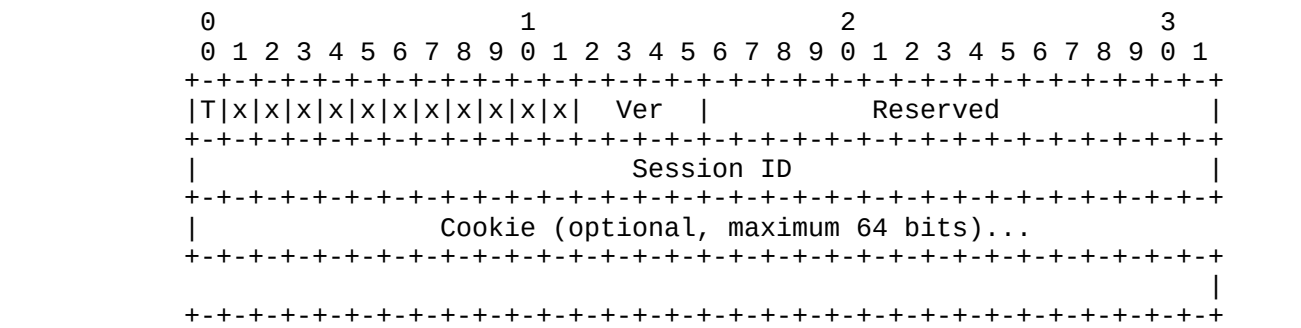
L2TPv3 over UDP must consider other L2 tunneling protocols that may be operating in the same environment, including L2TPv2 [RFC2661] and L2F [RFC2341].

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.

4.1.2.1. L2TP Session Header Over UDP

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

Figure 4.1.2.1: L2TPv3 Session Header over UDP



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.

Note that the initial bits 1, 4, 6, and 7 have meaning in L2TPv2 [RFC2661], and are deprecated and marked as reserved in L2TPv3. Thus, for UDP mode on a system that supports both versions of L2TP, it is important that the Ver field be inspected first to determine the Version of the header before acting upon any of these bits.

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

4.1.2.2. UDP Port Selection

The method for UDP Port Selection defined in this section is identical to that defined for L2TPv2 [RFC2661].

When negotiating a control connection over UDP, control messages 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.

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. A NAT device that can pass TFTP traffic with variant UDP ports 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

The tunneled frames that L2TP carry often have their own checksums or integrity checks, rendering the UDP checksum redundant for much of the L2TP data message contents. Thus, UDP checksums MAY be disabled in order to reduce the associated packet processing burden at the L2TP endpoints.

The L2TP header itself does not have its own checksum or integrity check. However, use of the L2TP Session ID and Cookie pair guards against accepting an L2TP data message if corruption of the Session ID or associated Cookie has occurred. When the L2-Specific Sublayer is present in the L2TP header, there is no built-in integrity check for the information contained therein if UDP checksums or some other integrity check is not employed. IPsec (see Section 4.1.3) may be used for strong integrity protection of the entire contents of L2TP data messages.

UDP checksums MUST be enabled for L2TP control messages.

4.1.3. L2TP and IPsec

The L2TP data channel does not provide cryptographic security of any kind. If the L2TP data channel operates over a public or untrusted IP network where privacy of the L2TP data is of concern or sophisticated attacks against L2TP are expected to occur, IPsec [RFC2401] MUST be made available to secure the L2TP traffic.

Either L2TP over UDP or L2TP over IP may be secured with IPsec. [RFC3193] defines the recommended method for securing L2TPv2. L2TPv3 possesses identical characteristics to IPsec as L2TPv2 when running over UDP and implementations MUST follow the same recommendation. When operating over IP directly, [RFC3193] still applies, though references to UDP source and destination ports (in particular, those

in Section 4, "IPsec Filtering details when protecting L2TP") may be ignored. Instead, the selectors used to identify L2TPv3 traffic are simply the source and destination IP addresses for the tunnel endpoints together with the L2TPv3 IP protocol type, 115.

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 may be performed at the network layer above L2TP. These network layer access control features may be handled at an LCCE via vendor-specific authorization features, 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.

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

4.1.4. IP Fragmentation Issues

Fragmentation and reassembly in network equipment generally require significantly greater resources than sending or receiving a packet as a single unit. As such, fragmentation and reassembly should be avoided whenever possible. Ideal solutions for avoiding fragmentation include proper configuration and management of MTU sizes among the Remote System, the LCCE, and the IP network, as well as adaptive measures that operate with the originating host (e.g., [RFC1191], [RFC1981]) to reduce the packet sizes at the source.

An LCCE MAY fragment a packet before encapsulating it in L2TP. For example, if an IPv4 packet arrives at an LCCE from a Remote System that, after encapsulation with its associated framing, L2TP, and IP, does not fit in the available path MTU towards its LCCE peer, the local LCCE may perform IPv4 fragmentation on the packet before tunnel encapsulation. This creates two (or more) L2TP packets, each

carrying an IPv4 fragment with its associated framing. This ultimately has the effect of placing the burden of fragmentation on the LCCE, while reassembly occurs on the IPv4 destination host.

If an IPv6 packet arrives at an LCCE from a Remote System that, after encapsulation with associated framing, L2TP and IP, does not fit in the available path MTU towards its L2TP peer, the Generic Packet Tunneling specification [RFC2473], Section 7.1 SHOULD be followed. In this case, the LCCE should either send an ICMP Packet Too Big message to the data source, or fragment the resultant L2TP/IP packet (for reassembly by the L2TP peer).

If the amount of traffic requiring fragmentation and reassembly is rather light, or there are sufficiently optimized mechanisms at the tunnel endpoints, fragmentation of the L2TP/IP packet may be sufficient for accommodating mismatched MTUs that cannot be managed by more efficient means. This method effectively emulates a larger MTU between tunnel endpoints and should work for any type of L2-encapsulated packet. Note that IPv6 does not support "in-flight" fragmentation of data packets. Thus, unlike IPv4, the MTU of the path towards an L2TP peer must be known in advance (or the last resort IPv6 minimum MTU of 1280 bytes utilized) so that IPv6 fragmentation may occur at the LCCE.

In summary, attempting to control the source MTU by communicating with the originating host, forcing that an MTU be sufficiently large on the path between LCCE peers to tunnel a frame from any other interface without fragmentation, fragmenting IP packets before encapsulation with L2TP/IP, or fragmenting the resultant L2TP/IP packet between the tunnel endpoints, are all valid methods for managing MTU mismatches. Some are clearly better than others depending on the given deployment. For example, a passive monitoring application using L2TP would certainly not wish to have ICMP messages sent to a traffic source. Further, if the links connecting a set of LCCEs have a very large MTU (e.g., SDH/SONET) and it is known that the MTU of all links being tunneled by L2TP have smaller MTUs (e.g., 1500 bytes), then any IP fragmentation and reassembly enabled on the participating LCCEs would never be utilized. An implementation MUST implement at least one of the methods described in this section for managing mismatched MTUs, based on careful consideration of how the final product will be deployed.

L2TP-specific fragmentation and reassembly methods, which may or may not depend on the characteristics of the type of link being tunneled (e.g., judicious packing of ATM cells), may be defined as well, but these methods are outside the scope of this document.

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 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 an ACK message.

All control messages take up one slot in the control message sequence number space, except the ACK message. Thus, Ns is not incremented after an ACK 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-ACK message received plus 1, modulo 65536). While the Nr in a received ACK message 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 ACK message. 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 but SHOULD be configurable) 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 SHOULD be no less than 8 seconds per retransmission. If no peer response is detected after several retransmissions (a recommended default is 10, but MUST be configurable), the control connection and all associated sessions MUST be cleared. As it is the first message to establish a control connection, the SCCRQ MAY employ a different retransmission maximum than other control messages in order to help facilitate failover to alternate LCCEs in a timely fashion.

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 StopCCN message has been sent (e.g., 1 + 2 + 4 + 8 + 8... seconds), or until the StopCCN message itself has been acknowledged.

A sliding window mechanism is used for control message transmission and retransmission. 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 a maximum of N outstanding (i.e., unacknowledged) 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 advertise a non-zero receive window as small or as large as it wishes, depending on its own ability to process incoming messages before sending an acknowledgement. Each peer MUST limit the number of unacknowledged messages it will send before receiving an acknowledgement by this Receive Window Size. The actual internal

unacknowledged message send-queue depth may be further limited by local resource allocation or by dynamic slow-start and congestion-avoidance mechanisms.

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 MAY drop messages, but MUST NOT actively delay acknowledgment of messages as a technique for flow control of control messages. Appendix B contains examples of control message transmission, acknowledgment, and retransmission.

4.3. Control Message Authentication

L2TP incorporates an optional authentication and integrity check for all control messages. This mechanism consists of a computed one-way hash over the header and body of the L2TP control message, a pre-configured shared secret, and a local and remote nonce (random value) exchanged via the Control Message Authentication Nonce AVP. This per-message authentication and integrity check is designed to perform a mutual authentication between L2TP nodes, perform integrity checking of all control messages, and guard against control message spoofing and replay attacks that would otherwise be trivial to mount.

At least one shared secret (password) MUST exist between communicating L2TP nodes to enable Control Message Authentication. See Section 5.4.3 for details on calculation of the Message Digest and construction of the Control Message Authentication Nonce and Message Digest AVPs.

L2TPv3 Control Message Authentication is similar to L2TPv2 [RFC2661] Tunnel Authentication in its use of a shared secret and one-way hash calculation. The principal difference is that, instead of computing the hash over selected contents of a received control message (e.g., the Challenge AVP and Message Type) as in L2TPv2, the entire message is used in the hash in L2TPv3. In addition, instead of including the hash digest in just the SCCRP and SCCCN messages, it is now included in all L2TP messages.

The Control Message Authentication mechanism is optional, and may be disabled if both peers agree. For example, if IPsec is already being used for security and integrity checking between the LCCEs, the function of the L2TP mechanism becomes redundant and may be disabled.

Presence of the Control Message Authentication Nonce AVP in an SCCRQ or SCCRP message serves as indication to a peer that Control Message Authentication is enabled. If an SCCRQ or SCCRP contains a Control Message Authentication Nonce AVP, the receiver of the message MUST

respond with a Message Digest AVP in all subsequent messages sent. Control Message Authentication is always bidirectional; either both sides participate in authentication, or neither does.

If Control Message Authentication is disabled, the Message Digest AVP still MAY be sent as an integrity check of the message. The integrity check is calculated as in Section 5.4.3, with an empty zero-length shared secret, local nonce, and remote nonce. If an invalid Message Digest is received, it should be assumed that the message has been corrupted in transit and the message dropped accordingly.

Implementations MAY rate-limit control messages, particularly SCCRQ messages, upon receipt for performance reasons or for protection against denial of service attacks.

4.4. Keepalive (Hello)

L2TP employs a keepalive mechanism to detect loss of connectivity between a pair of LCCEs. This is accomplished by injecting Hello control messages (see Section 6.5) after a 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.

Since 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.

Periodic keepalive for the control connection MUST be implemented by sending a Hello if a period of time (a recommended default is 60 seconds, but MUST 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 MUST employ a jittered timer mechanism to prevent grouping of Hello messages.

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

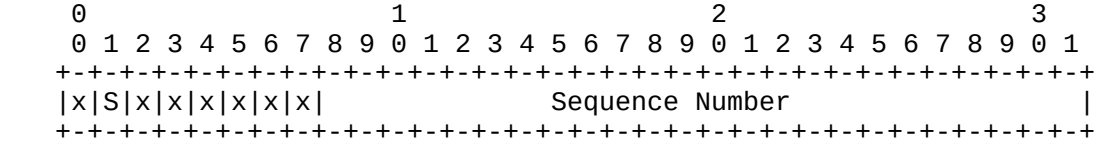
outgoing data message, the sender places the identifier specified in the Local Session ID AVP (received from peer during session establishment) in the Session ID field of the L2TP data header. In this manner, session frames are multiplexed and demultiplexed between a given pair of LCCEs. Multiple control 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. It is important for implementers to note that the Cookie field check occurs after looking up the session context by the Session ID, and as such, consists merely of a value match of the Cookie field and that stored in the retrieved context. There is no need to perform a lookup across the Session ID and Cookie as a single value. Any received data packets that contain invalid Session IDs or associated Cookie values MUST be dropped. Finally, the LCCE either forwards the network packet within the tunneled frame (e.g., as an LNS) or switches the frame to a circuit (e.g., as an LAC).

4.6. Default L2-Specific Sublayer

This document defines a Default L2-Specific Sublayer format (see Section 3.2.2) that a pseudowire may use for features such as sequencing support, L2 interworking, OAM, or other per-data-packet operations. The Default L2-Specific Sublayer 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 sublayers MAY be defined (e.g., an encapsulation with a larger Sequence Number field or timing information) and identified for use via the L2-Specific Sublayer Type AVP.

Figure 4.6: Default L2-Specific Sublayer Format



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, duplicate, or out-of-order packets within a given session.

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, duplicate, or silently drop packets. Reordering may break some non-IP protocols or L2 control traffic being carried by the link. Silent dropping or duplication of packets may break protocols that assume per-packet indications of error, such as TCP header compression. While a common mechanism for packet sequence detection is provided, 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 of data packets, packet duplication, or silent packet loss, sequencing may be enabled on some or all packets by using the S bit and Sequence Number field defined in the Default L2-Specific Sublayer (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 Data Sequencing AVP in Section 5.4.4).

When determining whether a packet is in or out of sequence, an implementation SHOULD utilize a method that is resilient to temporary dropouts in connectivity coupled with high per-session packet rates. The recommended method is outlined in Appendix C.

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 at least 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, without normative reference to RFC 2661.

If one wishes to tunnel PPP over L2TPv3, and fallback to L2TPv2 only if it is not available, then L2TPv3 over UDP with automatic fallback (see Section 4.7.3) MUST be used. There is no deterministic method for automatic fallback from L2TPv3 over IP to either L2TPv2 or L2TPv3 over UDP. One could infer whether L2TPv3 over IP is supported by sending an SCCRQ and waiting for a response, but this could be problematic during periods of packet loss between L2TP nodes.

4.7.2. L2TPv3 over UDP

The format of the L2TPv3 over UDP header is defined in Section 4.1.2.1.

When operating over UDP, L2TPv3 uses the same port (1701) as L2TPv2 and shares the first two octets of header format with L2TPv2. The Ver field is used to distinguish L2TPv2 packets from L2TPv3 packets. If an implementation is capable of operating in L2TPv2 or L2TPv3 modes, it is possible to automatically detect whether a peer can support L2TPv2 or L2TPv3 and operate accordingly. The details of this fallback capability is defined in the following section.

4.7.3. Automatic L2TPv2 Fallback

When running over UDP, an implementation may detect whether a peer is L2TPv3-capable by sending a special SCCRQ that is properly formatted for both L2TPv2 and L2TPv3. This is accomplished by sending an SCCRQ with its Ver field set to 2 (for L2TPv2), and ensuring that any L2TPv3-specific AVPs (i.e., AVPs present within this document and not defined within RFC 2661) in the message are sent with each M bit set to 0, and that all L2TPv2 AVPs are present as they would be for L2TPv2. This is done so that L2TPv3 AVPs will be ignored by an L2TPv2-only 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 in L2TPv3, and the 16-bit Tunnel ID and

16-bit Session ID in L2TPv2, are always 0 for an SCCRQ. This effectively hides the fact that there are a pair of 16-bit fields in L2TPv2, and a single 32-bit field in L2TPv3.

If the peer implementation is L2TPv3-capable, a control message with the Ver field set to 3 and an L2TPv3 header and message format will be sent in response to the SCCRQ. Operation may then continue as L2TPv3. If a message is received with the Ver field set to 2, it must be assumed that the peer implementation is L2TPv2-only, thus enabling fallback to L2TPv2 mode to safely occur.

Note Well: The L2TPv2/v3 auto-detection mode requires that all L2TPv3 implementations over UDP be liberal in accepting an SCCRQ control message with the Ver field set to 2 or 3 and the presence of L2TPv2-specific AVPs. An L2TPv3-only implementation MUST ignore all L2TPv2 AVPs (e.g., those defined in RFC 2661 and not in this document) within an SCCRQ with the Ver field set to 2 (even if the M bit is set on the L2TPv2-specific AVPs).

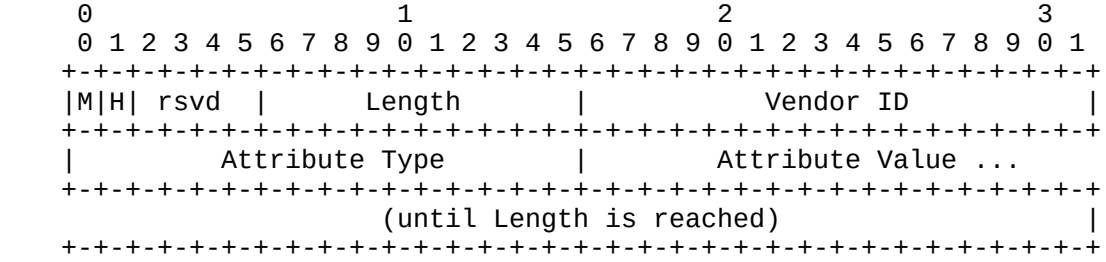
5. Control Message Attribute Value Pairs

To maximize extensibility while permitting interoperability, a uniform method for encoding message types 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 when sent and ignored upon receipt.

Mandatory (M) bit: Controls the behavior required of an implementation that receives an unrecognized AVP. The M bit of a given AVP MUST only be inspected and acted upon if the AVP is unrecognized (see Section 5.2).

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: Contains 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.

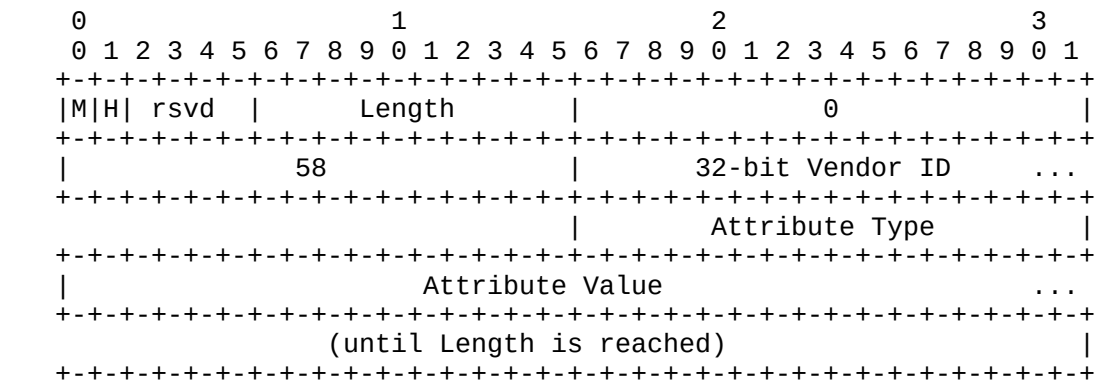
Vendor ID: The IANA-assigned "SMI Network Management Private Enterprise Codes" [RFC1700] value. The value 0, corresponding to IETF-adopted attribute values, is used for all AVPs defined within this document. Any vendor wishing to implement its own L2TP extensions can use its own Vendor ID along with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

In the event that the 16-bit Vendor ID space is exhausted, vendor-specific AVPs with a 32-bit Vendor ID MUST be encapsulated in the following manner:

Figure 5.2: Extended Vendor ID AVP Format



This AVP encodes a vendor-specific AVP with a 32-bit Vendor ID space within the Attribute Value field. Multiple AVPs of this type may exist in any message. The 16-bit Vendor ID MUST be 0, indicating that this is an IETF-defined AVP, and the Attribute Type MUST be 58, indicating that what follows is a vendor-specific AVP with a 32-bit Vendor ID code. This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP MUST be set to 0. The Length of the AVP is 12 plus the length of the Attribute Value.

5.2. Mandatory AVPs and Setting the M Bit

If the M bit is set on an AVP that is unrecognized by its recipient, the session or control connection associated with the control message containing the AVP MUST be shut down. If the control message containing the unrecognized AVP is associated with a session (e.g., an ICRQ, ICRP, ICCN, SLI, etc.), then the session MUST be issued a CDN with a Result Code of 2 and Error Code of 8 (as defined in Section 5.4.2) and shut down. If the control message containing the unrecognized AVP is associated with establishment or maintenance of a Control Connection (e.g., SCCRQ, SCCRP, SCCCN, Hello), then the associated Control Connection MUST be issued a StopCCN with Result Code of 2 and Error Code of 8 (as defined in Section 5.4.2) and shut down. If the M bit is not set on an unrecognized AVP, the AVP MUST be ignored when received, processing the control message as if the AVP were not present.

Receipt of an unrecognized 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 set for AVPs that are deemed crucial to proper operation of the session or control connection by the sender. AVPs that are considered crucial by the sender may vary by application and configured options. In no case shall a receiver of

an AVP "validate" if the M bit is set on a recognized AVP. If the AVP is recognized (as all AVPs defined in this document MUST be for a compliant L2TPv3 specification), then by definition, the M bit is of no consequence.

The sender of an AVP is free to set its M bit to 1 or 0 based on whether the configured application strictly requires the value contained in the AVP to be recognized or not. For example, "Automatic L2TPv2 Fallback" in Section 4.7.3 requires the setting of the M bit on all new L2TPv3 AVPs to zero if fallback to L2TPv2 is supported and desired, and 1 if not.

The M bit is useful as extra assurance for support of critical AVP extensions. However, more explicit methods may be available to determine support for a given feature rather than using the M bit alone. For example, if a new AVP is defined in a message for which there is always a message reply (i.e., an ICRQ, ICRP, SCCRQ, or SCCRP message), rather than simply sending an AVP in the message with the M bit set, availability of the extension may be identified by sending an AVP in the request message and expecting a corresponding AVP in a reply message. This more explicit method, when possible, is preferred.

The M bit also plays a role in determining whether or not a malformed or out-of-range value within an AVP should be ignored or should result in termination of a session or control connection (see Section 7.1 for more details).

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, IDs, or other vital information.

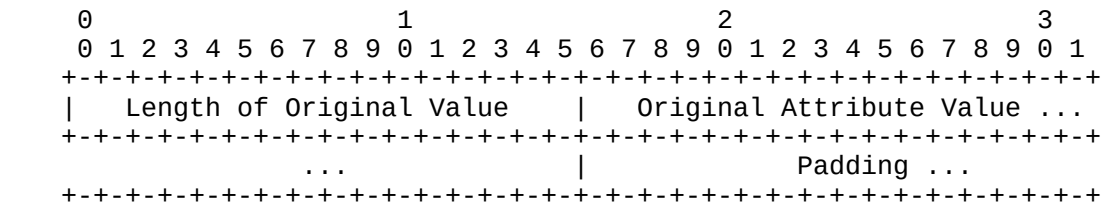
The H bit MUST only be set if (1) a shared secret exists between the LCCEs and (2) Control Message Authentication is enabled (see Section 4.3). If the H bit is set in any AVP(s) in a given control message, at least one Random Vector AVP must also be present in the message and MUST precede the first AVP having an H bit of 1.

The shared secret between LCCEs is used to derive a unique shared key for hiding and unhiding calculations. The derived shared key is obtained via an HMAC-MD5 keyed hash [RFC2104], with the key consisting of the shared secret, and with the data being hashed consisting of a single octet containing the value 1.

```
shared_key = HMAC_MD5 (shared_secret, 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 the Hidden AVP Subformat, which appears 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 would become 6 + Attribute Value length + size of the Length of Original Attribute Value field + Padding. Thus, if Padding is 12 octets, the AVP length would be 6 + 4 + 2 + 12 = 24 octets.

Next, an MD5 [RFC1321] hash is performed (in network byte order) on the concatenation of the following:

+ the 2-octet Attribute number of the AVP
+ the shared key
+ 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, but not for hiding two or more instances of an AVP with the same Attribute Type unless the Attribute Values in the two AVPs are also identical. When a different random vector is used for the hiding of subsequent AVPs, a new Random Vector AVP MUST be placed in the control 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 key 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 key 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 [RFC2865], 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 key S, the Random Vector RV, and the Attribute Type A. Break the value field into 16-octet chunks p_1, p_2, 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 b_1, b_2, etc.

```
b_1 = MD5 (A + S + RV)   c_1 = p_1 xor b_1
b_2 = MD5 (S + c_1)      c_2 = p_2 xor b_2
          .                        .
          .                        .
          .                        .
b_i = MD5 (S + c_i-1)    c_i = p_i xor b_i
```

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. General Control Message AVPs

Message Type (All Messages)

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

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

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

The Message Type is a 2-octet unsigned integer.

The Message Type AVP MUST be the first AVP in a message, immediately following the control message header (defined in Section 3.2.1). See Section 3.1 for the list of defined control message types and their identifiers.

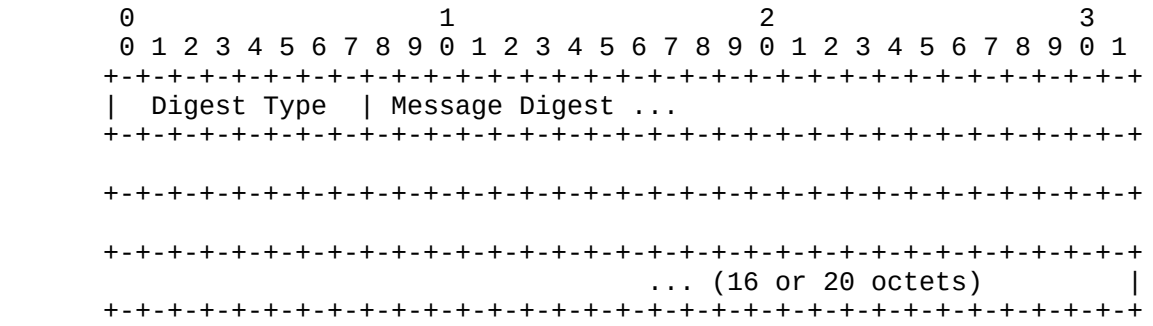
The Mandatory (M) bit within the Message Type AVP has special meaning. Rather than an indication as to whether the AVP itself should be ignored if not recognized, it is an indication as to whether the control message itself should be ignored. If the M bit is set within the Message Type AVP and the Message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

Message Digest (All Messages)

The Message Digest AVP, Attribute Type 59 is used as an integrity and authentication check of the L2TP Control Message header and body.

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



Digest Type is a one-octet integer indicating the Digest calculation algorithm:

0 HMAC-MD5 [RFC2104]
1 HMAC-SHA-1 [RFC2104]

Digest Type 0 (HMAC-MD5) MUST be supported, while Digest Type 1 (HMAC-SHA-1) SHOULD be supported.

The Message Digest is of variable length and contains the result of the control message authentication and integrity calculation. For Digest Type 0 (HMAC-MD5), the length of the digest MUST be 16

bytes. For Digest Type 1 (HMAC-SHA-1) the length of the digest MUST be 20 bytes.

If Control Message Authentication is enabled, at least one Message Digest AVP MUST be present in all messages and MUST be placed immediately after the Message Type AVP. This forces the Message Digest AVP to begin at a well-known and fixed offset. A second Message Digest AVP MAY be present in a message and MUST be placed directly after the first Message Digest AVP.

The shared secret between LCCEs is used to derive a unique shared key for Control Message Authentication calculations. The derived shared key is obtained via an HMAC-MD5 keyed hash [RFC2104], with the key consisting of the shared secret, and with the data being hashed consisting of a single octet containing the value 2.

```
shared_key = HMAC_MD5 (shared_secret, 2)
```

Calculation of the Message Digest is as follows for all messages other than the SCCRQ (where "+" refers to concatenation):

```
Message Digest = HMAC_Hash (shared_key, local_nonce +
                            remote_nonce + control_message)
```

HMAC_Hash: HMAC Hashing algorithm identified by the Digest Type (MD5 or SHA1)

local_nonce: Nonce chosen locally and advertised to the remote LCCE.

remote_nonce: Nonce received from the remote LCCE

(The local_nonce and remote_nonce are advertised via the Control Message Authentication Nonce AVP, also defined in this section.)

shared_key: Derived shared key for this control connection

control_message: The entire contents of the L2TP control message, including the control message header and all AVPs. Note that the control message header in this case begins after the all-zero Session ID when running over IP (see Section 4.1.1.2), and after the UDP header when running over UDP (see Section 4.1.2.1).

When calculating the Message Digest, the Message Digest AVP MUST be present within the control message with the Digest Type set to its proper value, but the Message Digest itself set to zeros.

When receiving a control message, the contents of the Message Digest AVP MUST be compared against the expected digest value based on local calculation. This is done by performing the same digest calculation above, with the local_nonce and remote_nonce reversed. This message authenticity and integrity checking MUST be performed before utilizing any information contained within the control message. If the calculation fails, the message MUST be dropped.

The SCCRQ has special treatment as it is the initial message commencing a new control connection. As such, there is only one nonce available. Since the nonce is present within the message itself as part of the Control Message Authentication Nonce AVP, there is no need to use it in the calculation explicitly. Calculation of the SCCRQ Message Digest is performed as follows:

```
Message Digest = HMAC_Hash (shared_key, control_message)
```

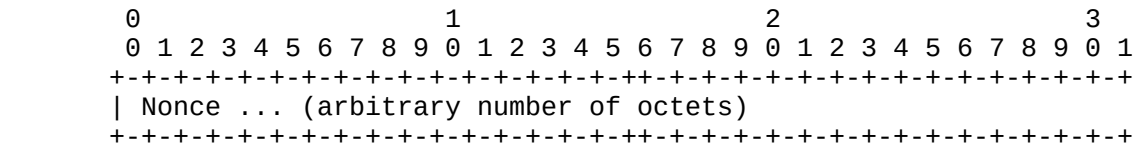
To allow for graceful switchover to a new shared secret or hash algorithm, two Message Digest AVPs MAY be present in a control message, and two shared secrets MAY be configured for a given LCCE. If two Message Digest AVPs are received in a control message, the message MUST be accepted if either Message Digest is valid. If two shared secrets are configured, each (separately) MUST be used for calculating a digest to be compared to the Message Digest(s) received. When calculating a digest for a control message, the Value field for both of the Message Digest AVPs MUST be set to zero.

This AVP MUST NOT be hidden (the H bit MUST be 0). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). The Length is 23 for Digest Type 1 (HMAC-MD5), and 27 for Digest Type 2 (HMAC-SHA-1).

Control Message Authentication Nonce (SCCRQ, SCCRP)

The Control Message Authentication Nonce AVP, Attribute Type 73, MUST contain a cryptographically random value [RFC1750]. This value is used for Control Message Authentication.

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



The Nonce is of arbitrary length, though at least 16 octets is recommended. The Nonce contains the random value for use in the Control Message Authentication hash calculation (see Message Digest AVP definition in this section).

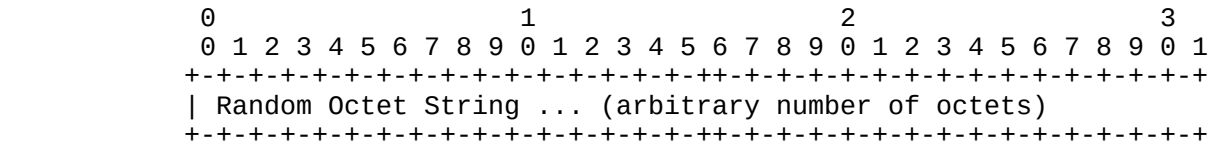
If Control Message Authentication is enabled, this AVP MUST be present in the SCCRQ and SCCRP messages.

This AVP MUST NOT be hidden (the H bit MUST be 0). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). The Length of this AVP is 6 plus the length of the Nonce.

Random Vector (All Messages)

The Random Vector AVP, Attribute Type 36, MUST contain a cryptographically random value [RFC1750]. This value is used for AVP Hiding.

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



The Random Octet String is of arbitrary length, though 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. As such, at least one Random Vector AVP MUST precede the first AVP with the H bit set.

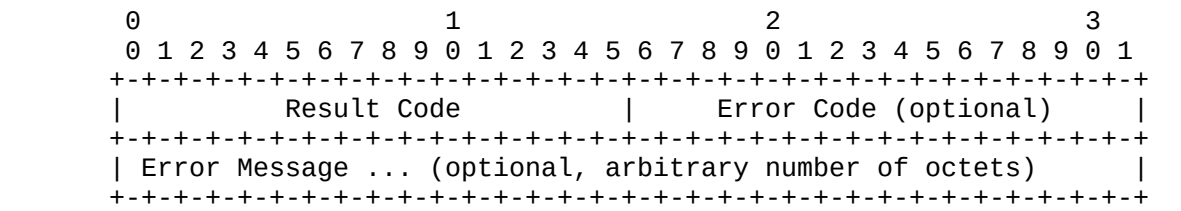
This AVP MUST NOT be hidden (the H bit MUST be 0). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). The Length of this AVP is 6 plus the length of the Random Octet String.

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 connection 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 [RFC3629] 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, but MAY vary (see Section 5.2). 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 plus 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 connection already exists.
- 4 - Requester is not authorized to establish a control connection.
- 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 or timeout

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).
13 - Session not established due to losing tie breaker.
14 - Session not established due to unsupported PW type.
15 - Session not established, sequencing required without valid L2-Specific Sublayer.
16 - Finite state machine error or timeout.

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., "192.0.2.1, 192.0.2.2, 192.0.2.3") in the UTF-8 charset [RFC3629] using the Default Language [RFC2277]. If there are no servers for the LAC or LNS to suggest, then Error Code 7 should be used. For IPv4, the delimiter between addresses MUST be precisely a single comma and a single space. For IPv6, each literal address MUST be enclosed in "[" and "]" characters, following the encoding described in [RFC2732].

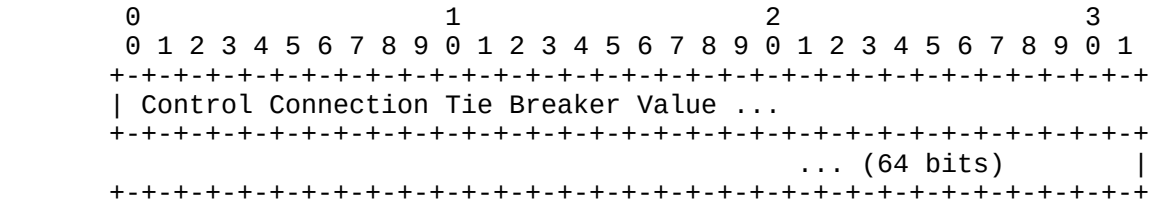
When a General Error Code of 6 is used, additional information about the error SHOULD be included in the Error Message field. A vendor-specific AVP MAY be sent to more precisely detail a vendor-specific problem.

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; if so, a tie has been detected. In this case, the LCCE must compare its Control Connection Tie Breaker value with the one received in the SCCRQ. The lower value "wins", and the "loser" MUST discard its control connection. A StopCCN SHOULD be sent by the winner as an explicit rejection for the losing SCCRQ. 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.

Applications that employ a distinct and well-known initiator have no need for tie breaking, and MAY omit this AVP or disable tie breaking functionality. Applications that require tie breaking also require that an LCCE be uniquely identifiable upon receipt of an SCCRQ. For L2TP over IP, this MUST be accomplished via the Router ID AVP.

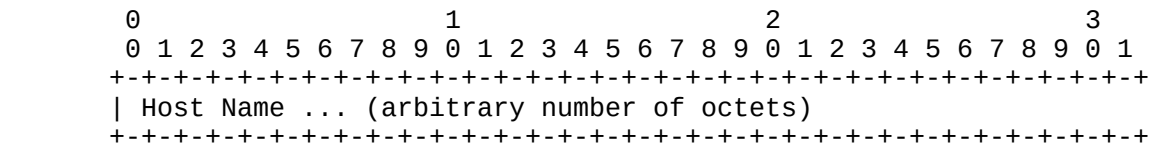
Note that in [RFC2661], this AVP is referred to as the "Tie Breaker AVP" and is applicable only to a control connection. In L2TPv3, the AVP serves the same purpose of tie breaking, but is applicable to a control connection or a session. The Control Connection Tie Breaker AVP (present only in Control Connection messages) and Session Tie Breaker AVP (present only in Session messages), are described separately in this document, but share the same Attribute type of 5.

This AVP MUST NOT be hidden (the H bit MUST be 0). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). 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, encoded in the US-ASCII charset.

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.

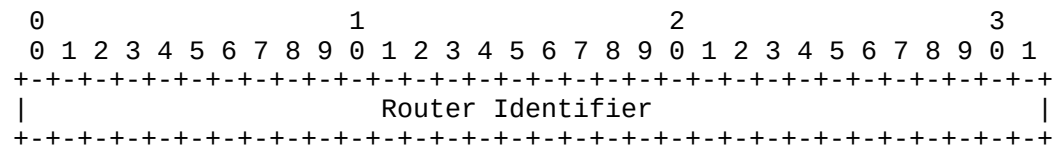
This name should be as broadly unique as possible; for hosts participating in DNS [RFC1034], a host name with fully qualified domain would be appropriate. The Host Name AVP and/or Router ID AVP MUST be used to identify an LCCE as described in Section 3.3.

This AVP MUST NOT be hidden (the H bit MUST be 0). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). The Length of this AVP is 6 plus the length of the Host Name.

Router ID (SCCRQ, SCCRP)

The Router ID AVP, Attribute Type 60, is an identifier used to identify an LCCE for control connection setup, tie breaking, and/or tunnel authentication.

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



The Router Identifier is a 4-octet unsigned integer. Its value is unique for a given LCCE, per Section 8.1 of [RFC2072]. The Host Name AVP and/or Router ID AVP MUST be used to identify an LCCE as described in Section 3.3.

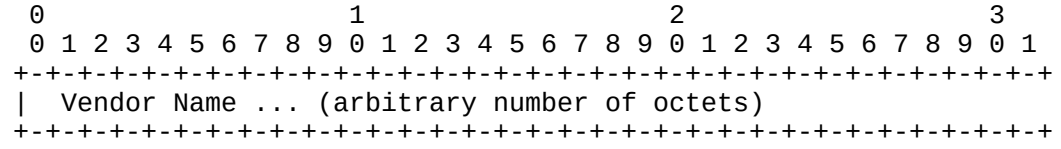
Implementations MUST NOT assume that Router Identifier is a valid IP address. The Router Identifier for L2TP over IPv6 can be obtained from an IPv4 address (if available) or via unspecified implementation-specific means.

This AVP MUST NOT be hidden (the H bit MUST be 0). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). The Length of this AVP is 10.

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:



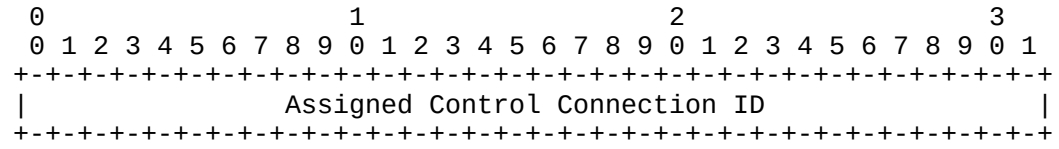
The Vendor Name is the indicated number of octets representing the vendor string. Human-readable text for this AVP MUST be provided in the US-ASCII charset [RFC1958, RFC2277].

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 0, but MAY vary (see Section 5.2). 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 61, contains 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.

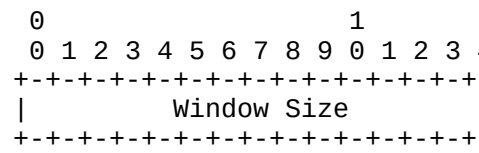
Under certain circumstances, an LCCE may need to send a StopCCN to a peer without having yet received an Assigned Control Connection ID AVP from the peer (i.e., SCCRQ sent, no SCCRP received yet). In this case, the Assigned Control Connection ID AVP that had been sent to the peer earlier (i.e., in the 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, but MAY vary (see Section 5.2). 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.

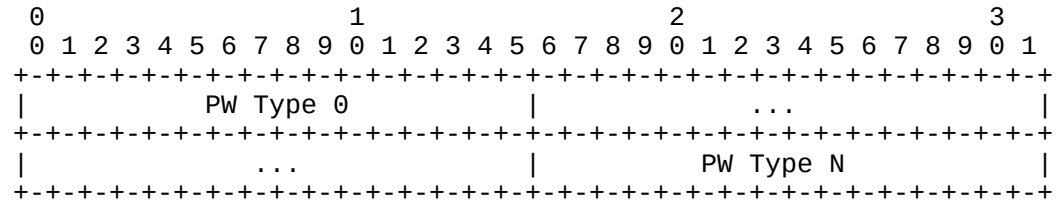
The remote peer may send the specified number of control messages before it must wait for an acknowledgment. See Section 4.2 for more information on reliable control message delivery.

This AVP MUST NOT be hidden (the H bit MUST be 0). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). The Length of this AVP is 8.

Pseudowire Capabilities List (SCCRQ, SCCRP)

The Pseudowire Capabilities List (PW Capabilities List) AVP, Attribute Type 62, 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 managed by IANA and will appear in associated pseudowire-specific documents for each PW type.

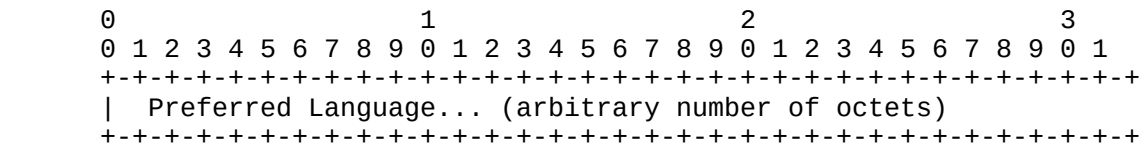
If a sender includes a given PW type in the PW Capabilities List AVP, the sender assumes full responsibility for supporting that particular payload, such as any payload-specific AVPs, L2-Specific Sublayer, or control messages that may be defined in the appropriate companion document.

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). The Length (before hiding) of this AVP is 8 octets with one PW type specified, plus 2 octets for each additional PW type.

Preferred Language (SCCRQ, SCCRP)

The Preferred Language AVP, Attribute Type 72, provides a method for an LCCE to indicate to the peer the language in which human-readable messages it sends SHOULD be composed. This AVP contains a single language tag or language range [RFC3066].

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



The Preferred Language is the indicated number of octets representing the language tag or language range, encoded in the US-ASCII charset.

It is not required to send a Preferred Language AVP. If (1) an LCCE does not signify a language preference by the inclusion of this AVP in the SCCRQ or SCCRP, (2) the Preferred Language AVP is unrecognized, or (3) the requested language is not supported by the peer LCCE, the default language [RFC2277] MUST be used for all internationalized strings sent by the peer.

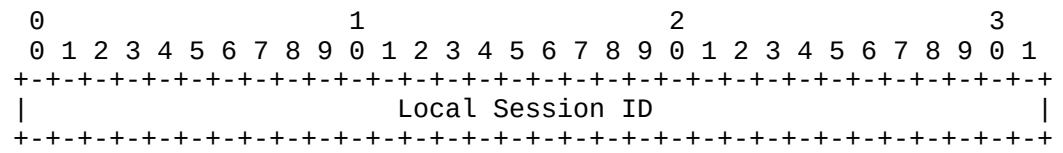
This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 0, but MAY vary (see Section 5.2). The Length (before hiding) of this AVP is 6 plus the length of the Preferred Language.

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 63, contains 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 two identifiers used to multiplex and demultiplex sessions between two LCCEs. Each LCCE chooses any free value it desires, and sends it to the remote LCCE using this AVP. The remote LCCE MUST then send all data packets associated with this session using this value. Additionally, for all session-oriented control messages sent after this AVP is received (e.g., ICRP, ICCN, CDN, SLI, etc.), the remote LCCE MUST echo this value in the Remote Session ID AVP.

Note that a Session ID value is unidirectional. Because each LCCE chooses its Session ID independent of its peer LCCE, the value does not have to match in each direction for a given session.

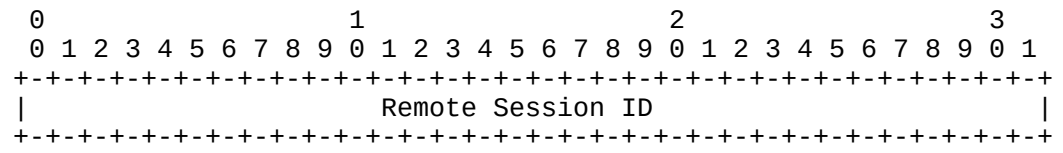
See Section 4.1 for additional information about the Session ID.

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be 1 set to 1, but MAY vary (see Section 5.2). 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 64, contains 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.

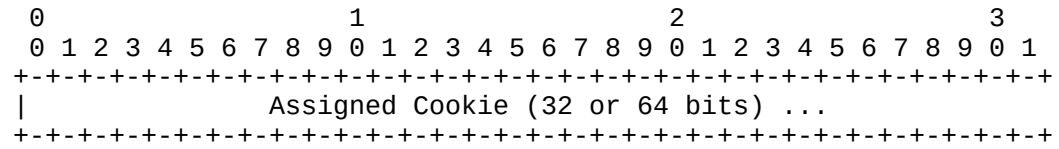
When a session-level control message must be sent to the peer before the Local Session ID AVP has been received, the value of the Remote Session ID AVP MUST be set to zero. Additionally, the Local Session ID AVP (sent in a previous control message for this session) MUST be included in the control message. 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 SHOULD be set to 1, but MAY vary (see Section 5.2). The Length (before hiding) of this AVP is 10.

Assigned Cookie (ICRQ, ICRP, OCRQ, OCRP)

The Assigned Cookie AVP, Attribute Type 65, contains 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 missing Assigned Cookie AVP or Assigned Cookie Value of zero length indicates that the Cookie field should not be present in any data packets sent to the LCCE sending this AVP.

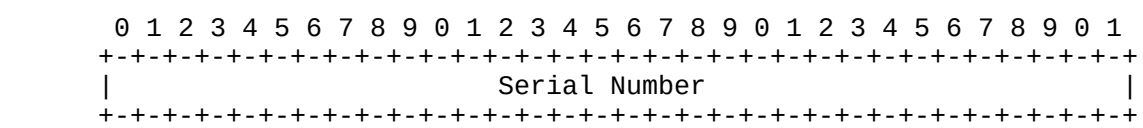
See Section 4.1 for additional information about the Assigned Cookie.

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). The Length (before hiding) of this AVP may be 6, 10, or 14 octets.

Serial Number (ICRQ, OCRQ)

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

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



The Serial Number is a 32-bit value.

The 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. 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.

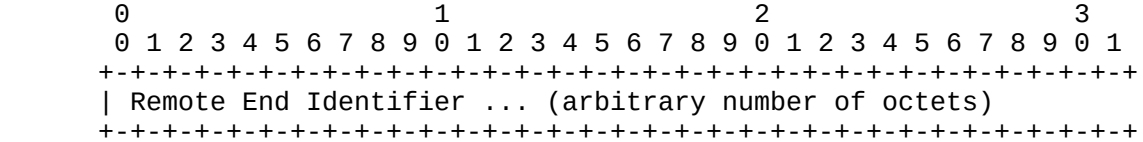
Note that in RFC 2661, this value was referred to as the "Call Serial Number AVP". It serves the same purpose and has the same attribute value and composition.

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 0, but MAY vary (see Section 5.2). The Length (before hiding) of this AVP is 10.

Remote End ID (ICRQ, OCRQ)

The Remote End ID AVP, Attribute Type 66, contains an identifier used to bind L2TP sessions to a given circuit, interface, or bridging instance. It also may be used to detect session-level ties.

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



The Remote End Identifier field is a variable-length field whose value is unique for a given LCCE peer, as described in Section 3.3.

A session-level tie is detected if an LCCE receives an ICRQ or OCRQ with an End ID AVP whose value matches that which 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 Session Tie Breaker AVP.

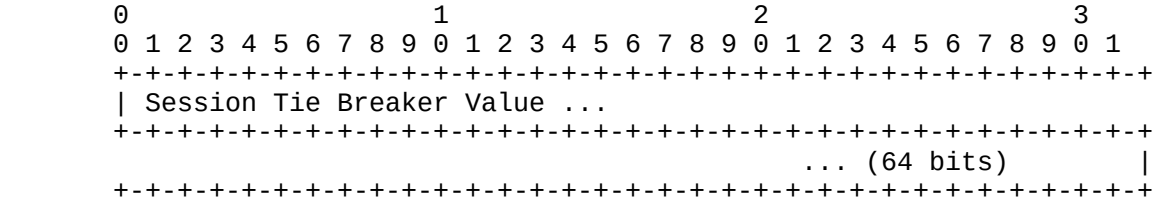
By default, the LAC-LAC cross-connect application (see Section 2(b)) of L2TP over an IP network MUST utilize the Router ID AVP and Remote End ID AVP to associate a circuit to an L2TP session. Other AVPs MAY be used for LCCE or circuit identification as specified in companion documents.

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). The Length (before hiding) of this AVP is 6 plus the length of the Remote End Identifier value.

Session Tie Breaker (ICRQ, OCRQ)

The Session Tie Breaker AVP, Attribute Type 5, 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. A tie is detected by examining the peer's identity (described in Section 3.3) plus the per-session shared value communicated via the End ID AVP. In the case of a tie, the recipient of an ICRQ or OCRQ must compare the received tie breaker value with the one that it sent earlier. The LCCE with the lower value "wins" and MUST send a CDN with result code set to 13 (as defined in Section 5.4.2) in response to the losing ICRQ or OCRQ. In the case in which a tie is detected, tie

breakers are sent by both sides, and the tie breaker values are equal, both sides MUST discard their sessions and restart session negotiation with new random tie breaker values.

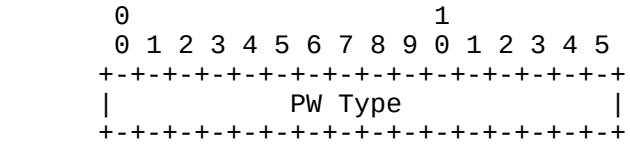
If a tie is detected but only one side sends a Session Tie Breaker AVP, the session initiator that included the Session Tie Breaker AVP "wins". If neither side issues a tie breaker, then both sides MUST tear down the session.

This AVP MUST NOT be hidden (the H bit MUST be 0). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). The Length of this AVP is 14.

Pseudowire Type (ICRQ, OCRQ)

The Pseudowire Type (PW Type) AVP, Attribute Type 68, 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:

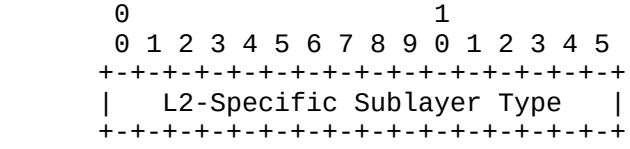


A peer MUST NOT request an incoming or outgoing call with a PW Type AVP specifying a value not advertised in the PW Capabilities List AVP it received during control connection establishment. Attempts to do so MUST result in the call being rejected via a CDN with the Result Code set to 14 (see Section 5.4.2).

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). The Length (before hiding) of this AVP is 8.

L2-Specific Sublayer (ICRQ, ICRP, ICCN, OCRQ, OCRP, OCCN)

The L2-Specific Sublayer AVP, Attribute Type 69, indicates the presence and format of the L2-Specific Sublayer the sender of this AVP requires on all incoming data packets for this L2TP session.



The L2-Specific Sublayer Type is a 2-octet unsigned integer with the following values defined in this document:

- 0 - There is no L2-Specific Sublayer present.
- 1 - The Default L2-Specific Sublayer (defined in Section 4.6) is used.

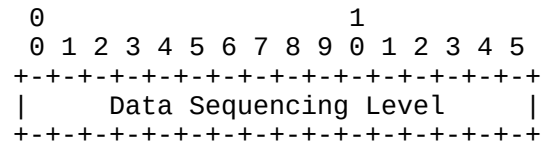
If this AVP is received and has a value other than zero, the receiving LCCE MUST include the identified L2-Specific Sublayer in its outgoing data messages. If the AVP is not received, it is assumed that there is no sublayer present.

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). The Length (before hiding) of this AVP is 8.

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

The Data Sequencing AVP, Attribute Type 70, indicates that the sender requires some or all of the data packets that it receives 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.

Defined Data Sequencing Levels are as follows:

- 0 - No incoming data packets require sequencing.
- 1 - Only non-IP data packets require sequencing.
- 2 - All incoming data packets 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 no Data Sequencing AVP is received, a Data Sequencing Level of 0 is assumed.

If a Data Sequencing Level of 1 is specified, only non-IP traffic carried within the tunneled L2 frame should have sequence numbers applied. Non-IP traffic here refers to any packets that cannot be

classified as an IP packet within their respective L2 framing (e.g., a PPP control packet or NETBIOS frame encapsulated by Frame Relay before being tunneled). All traffic that can be classified as IP MUST be sent with no sequencing (i.e., the S bit in the L2-Specific Sublayer is set to zero). If a packet is unable to be classified at all (e.g., because it has been compressed or encrypted at layer 2) or if an implementation is unable to perform such classification within L2 frames, all packets MUST be provided with sequence numbers (essentially falling back to a Data Sequencing Level of 2).

If a Data Sequencing Level of 2 is specified, all traffic MUST be sequenced.

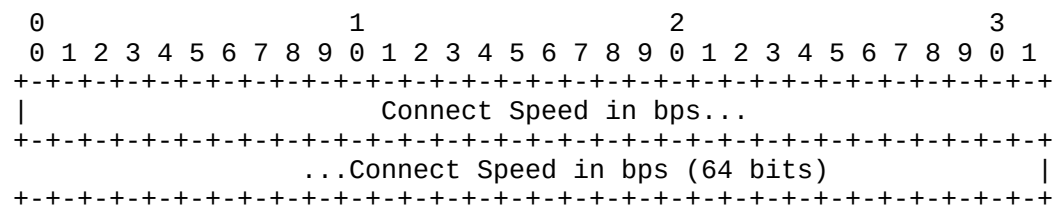
Data sequencing may only be requested when there is an L2-Specific Sublayer present that can provide sequence numbers. If sequencing is requested without requesting a L2-Specific Sublayer AVP, the session MUST be disconnected with a Result Code of 15 (see Section 5.4.2).

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). The Length (before hiding) of this AVP is 8.

Tx Connect Speed (ICRQ, ICRP, ICCN, OCRQ, OCRP, OCCN)

The Tx Connect Speed BPS AVP, Attribute Type 74, contains 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 an 8-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 (i.e., 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

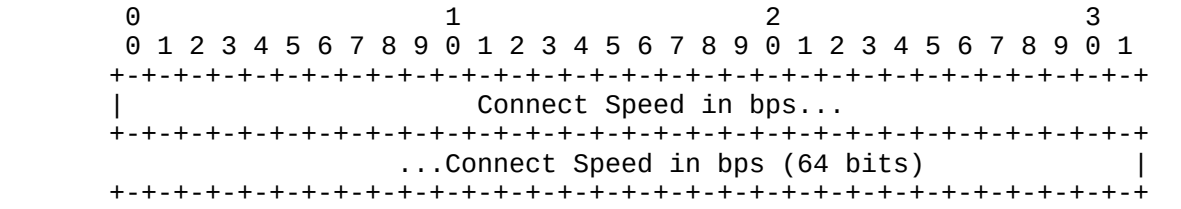
assumed to be symmetric and is represented by the single value in this AVP.

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 0, but MAY vary (see Section 5.2). The Length (before hiding) of this AVP is 14.

Rx Connect Speed (ICRQ, ICRP, ICCN, OCRQ, OCRP, OCCN)

The Rx Connect Speed AVP, Attribute Type 75, represents the speed of the connection from the perspective of the LAC (i.e., data flowing from the remote system to the LAC).

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



Connect Speed BPS is an 8-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.

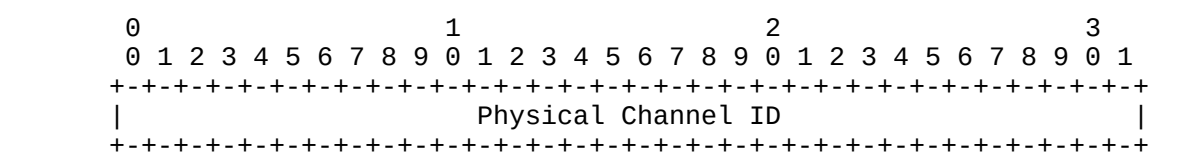
Presence of this AVP implies that the connection speed may be asymmetric with respect to the transmit connect speed given in the Tx Connect Speed AVP.

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 0, but MAY vary (see Section 5.2). The Length (before hiding) of this AVP is 14.

Physical Channel ID (ICRQ, ICRP, OCRP)

The Physical Channel ID AVP, Attribute Type 25, contains 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.

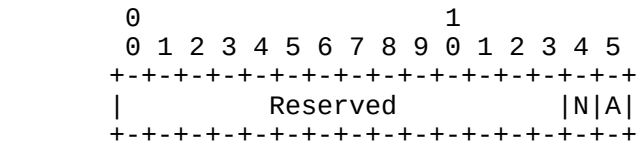
This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 0, but MAY vary (see Section 5.2). The Length (before hiding) of this AVP is 10.

5.4.5. Circuit Status AVPs

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

The Circuit Status AVP, Attribute Type 71, 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). Links that have a similar mechanism available (e.g., Frame Relay) MUST map the setting of this bit to the associated signaling for that link. Otherwise, the New bit SHOULD still be set the first time the L2TP session is established after provisioning.

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. Different circuit types have different names for status types. For example, 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 for the L2TP session.

In the simplest case, the circuit to which this AVP refers is a single physical interface, port, or circuit, depending on the application and the session setup. The status indication in this AVP may then be used to provide simple ILMI interworking for a variety of circuit types. For virtual or multipoint interfaces, the Circuit Status AVP is still utilized, but in this case, it refers to the state of an internal structure or a logical set of circuits. Each PW-specific companion document MUST specify precisely how this AVP is translated for each circuit type.

If this AVP is received with a Not Active notification for a given L2TP session, all data traffic for that session MUST cease (or not begin) in the direction of the sender of the Circuit Status AVP until the circuit is advertised as Active.

The Circuit Status MUST be advertised by this AVP in ICRQ, ICRP, OCRQ, and OCRP messages. Often, the circuit type will be marked Active when initiated, but subsequently MAY be advertised as Inactive. This indicates 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.

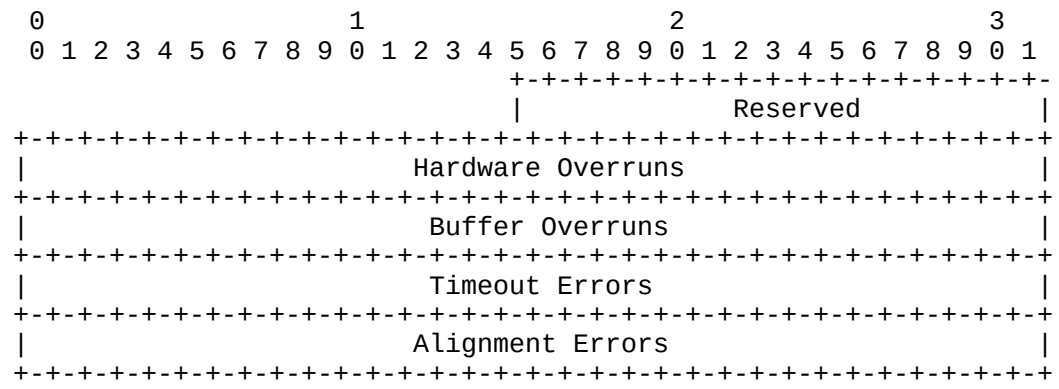
If additional circuit status information is needed for a given PW type, any new PW-specific AVPs MUST be defined in a separate document. This AVP is only for general circuit status information generally applicable to all circuit/interface types.

This AVP MAY be hidden (the H bit MAY be 0 or 1). The M bit for this AVP SHOULD be set to 1, but MAY vary (see Section 5.2). 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 0, but MAY vary (see Section 5.2). 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 packets 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
- Host Name
- Router ID
- Assigned Control Connection ID
- Pseudowire Capabilities List

The following AVPs MAY be present in the SCCRQ:

- Random Vector
- Control Message Authentication Nonce
- Message Digest
- Control Connection Tie Breaker
- Vendor Name
- Receive Window Size
- Preferred Language

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 that establishment of the control connection should continue.

The following AVPs MUST be present in the SCCRP:

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

The following AVPs MAY be present in the SCCRP:

- Random Vector
- Control Message Authentication Nonce
- Message Digest
- Vendor Name
- Receive Window Size
- Preferred Language

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:

- Random Vector
- Message Digest

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 AVPs MAY be present in the StopCCN:

- Random Vector
- Message Digest
- 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

The following AVP MAY be present in the HELLO:

- Random Vector
- Message Digest

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
- Serial Number
- Pseudowire Type
- Remote End ID
- Circuit Status

The following AVPs MAY be present in the ICRQ:

- Random Vector
- Message Digest
- Assigned Cookie
- Session Tie Breaker
- L2-Specific Sublayer
- Data Sequencing
- Tx Connect Speed
- Rx Connect Speed
- Physical Channel ID

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
- Circuit Status

The following AVPs MAY be present in the ICRP:

- Random Vector
- Message Digest
- Assigned Cookie
- L2-Specific Sublayer
- Data Sequencing
- Tx Connect Speed
- Rx Connect Speed
- Physical Channel ID

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 was 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:

- Random Vector
- Message Digest
- L2-Specific Sublayer
- Data Sequencing
- Tx Connect Speed
- Rx Connect Speed
- Circuit Status

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
- Serial Number
- Pseudowire Type
- Remote End ID
- Circuit Status

The following AVPs MAY be present in the OCRQ:

- Random Vector
- Message Digest
- Assigned Cookie
- Tx Connect Speed
- Rx Connect Speed
- Session Tie Breaker
- L2-Specific Sublayer
- Data Sequencing

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, which indicates that the call has been placed.

The following AVPs MUST be present in the OCRP:

- Message Type
- Local Session ID
- Remote Session ID
- Circuit Status

The following AVPs MAY be present in the OCRP:

- Random Vector
- Message Digest
- Assigned Cookie
- L2-Specific Sublayer
- Tx Connect Speed
- Rx Connect Speed
- Data Sequencing
- 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:

- Random Vector
- Message Digest
- L2-Specific Sublayer
- Tx Connect Speed
- Rx Connect Speed
- 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
- Result Code
- Local Session ID
- Remote Session ID

The following AVP MAY be present in the CDN:

- Random Vector
- Message Digest

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
- Local Session ID
- Remote Session ID
- Circuit Errors

The following AVP MAY be present in the WEN:

- Random Vector
- Message Digest

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 at an LNS or between an LAC and a Remote System, or if a forwarded Frame Relay VC transitions to Active or Inactive at an LAC, 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 pseudowire-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:

- Random Vector
- Message Digest
- Circuit Status

6.15. Explicit-Acknowledgement (ACK)

The Explicit Acknowledgement (ACK) message is used only to acknowledge receipt of a message or messages on the control connection (e.g., for purposes of updating Ns and Nr values). Receipt of this message does not trigger an event for the L2TP protocol state machine.

A message received without any AVPs (including the Message Type AVP), is referred to as a Zero Length Body (ZLB) message, and serves the same function as the Explicit Acknowledgement. ZLB messages are only permitted when Control Message Authentication defined in Section 4.3 is not enabled.

The following AVPs MAY be present in the ACK message:

Message Type
Message Digest

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 AVPs and 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.

When possible, a malformed AVP should be treated as an unrecognized AVP (see Section 5.2). Thus, an attempt to inspect the M bit SHOULD be made to determine the importance of the malformed AVP, and thus, the severity of the malformation to the entire control message. If the M bit can be reasonably inspected within the malformed AVP and is determined to be set, then as with an unrecognized AVP, the associated session or control connection MUST be shut down. If the M bit is inspected and is found to be 0, the AVP MUST be ignored (assuming recovery from the AVP malformation is indeed possible).

This policy must not be considered as a license to send malformed AVPs, but rather, as 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. One example of a malformed AVP situation that should be recoverable

is if the Rx Connect Speed AVP is received with a length of 10 rather than 14, implying that the connect speed bits-per-second is being formatted in 4 octets rather than 8. If the AVP does not have its M bit set (as would typically be the case), this condition is not considered catastrophic. As such, the control message should be accepted as though the AVP were not present (though a local error message may be 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. 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	Send StopCCN, 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, SCCRQ acceptable	Send SCCRP, Clean up losing connection	wait-ctl-conn
wait-ctl-reply	Receive SCCRQ, lose tie breaker, SCCRQ unacceptable	Send StopCCN, Clean up losing connection	idle
wait-ctl-reply	Receive SCCRQ, win tie breaker	Send StopCCN for losing connection	wait-ctl-reply
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 SCCRQ, SCCRP	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. If the SCCCN is valid, the control connection is established; otherwise, it is torn down (sending a StopCCN with the proper result and/or error code).

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.

7.3. 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.

7.3.1. 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, lose tie breaker	Process as ICRQ Recipient (Section 7.3.2)	idle
wait-reply	Receive ICRQ, win tie breaker	Send CDN for losing session	wait-reply
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-conn
: 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.

7.3.2. 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 established	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.

7.4. 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.

7.4.1. 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	Send CDN, clean up	idle
wait-reply	Receive OCRQ, lose tie breaker	Process as OCRQ Recipient (Section 7.4.2)	idle
wait-reply	Receive OCRQ, win tie breaker	Send CDN for losing session	wait-reply
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.

7.4.2. 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
wait-cs-answer, 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.

7.5. 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 full control message retransmission cycle (e.g., 1 + 2 + 4 + 8 ... seconds) 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 and Message Security

If a shared secret (password) exists between two LCCEs, it may be used to perform a mutual authentication between the two LCCEs, and construct an authentication and integrity check of arriving L2TP control messages. The mechanism provided by L2TPv3 is described in Section 4.3 and in the definition of the Message Digest and Control Message Authentication Nonce AVPs in Section 5.4.1.

This control message security mechanism provides for (1) mutual endpoint authentication, and (2) individual control message integrity and authenticity checking. Mutual endpoint authentication ensures that an L2TPv3 control connection is only established between two endpoints that are configured with the proper password. The individual control message and integrity check guards against accidental or intentional packet corruption (i.e., those caused by a control message spoofing or man-in-the-middle attack).

The shared secret that is used for all control connection, control message, and AVP security features defined in this document never needs to be sent in the clear between L2TP tunnel endpoints.

8.2. Data Packet Spoofing

Packet spoofing for any type of Virtual Private Network (VPN) protocol is of particular concern as insertion of carefully constructed rogue packets into the VPN transit network could result in a violation of VPN traffic separation, leaking data into a customer VPN. This is complicated by the fact that it may be particularly difficult for the operator of the VPN to even be aware that it has become a point of transit into or between customer VPNs.

L2TPv3 provides traffic separation for its VPNs via a 32-bit Session ID in the L2TPv3 data header. When present, the L2TPv3 Cookie (described in Section 4.1), provides an additional check to ensure that an arriving packet is intended for the identified session. Thus, use of a Cookie with the Session ID provides an extra guarantee that the Session ID lookup was performed properly and that the Session ID itself was not corrupted in transit.

In the presence of a blind packet spoofing attack, the Cookie may also provide security against inadvertent leaking of frames into a customer VPN. To illustrate the type of security that it is provided in this case, consider comparing the validation of a 64-bit Cookie in the L2TPv3 header to the admission of packets that match a given source and destination IP address pair. Both the source and destination IP address pair validation and Cookie validation consist of a fast check on cleartext header information on all arriving packets. However, since L2TPv3 uses its own value, it removes the requirement for one to maintain a list of (potentially several) permitted or denied IP addresses, and moreover, to guard knowledge of the permitted IP addresses from hackers who may obtain and spoof them. Further, it is far easier to change a compromised L2TPv3 Cookie than a compromised IP address," and a cryptographically random [RFC1750] value is far less likely to be discovered by brute-force attacks compared to an IP address.

For protection against brute-force, blind, insertion attacks, a 64-bit Cookie MUST be used with all sessions. A 32-bit Cookie is vulnerable to brute-force guessing at high packet rates, and as such, should not be considered an effective barrier to blind insertion attacks (though it is still useful as an additional verification of a successful Session ID lookup). The Cookie provides no protection against a sophisticated man-in-the-middle attacker who can sniff and correlate captured data between nodes for use in a coordinated attack.

The Assigned Cookie AVP is used to signal the value and size of the Cookie that must be present in all data packets for a given session. Each Assigned Cookie MUST be selected in a cryptographically random manner [RFC1750] such that a series of Assigned Cookies does not provide any indication of what a future Cookie will be.

The L2TPv3 Cookie must not be regarded as a substitute for security such as that provided by IPsec when operating over an open or untrusted network where packets may be sniffed, decoded, and correlated for use in a coordinated attack. See Section 4.1.3 for more information on running L2TP over IPsec.

9. Internationalization Considerations

The Host Name and Vendor Name AVPs are not internationalized. The Vendor Name AVP, although intended to be human-readable, would seem to fit in the category of "globally visible names" [RFC2277] and so is represented in US-ASCII.

If (1) an LCCE does not signify a language preference by the inclusion of a Preferred Language AVP (see Section 5.4.3) in the

SCCRQ or SCCRP, (2) the Preferred Language AVP is unrecognized, or (3) the requested language is not supported by the peer LCCE, the default language [RFC2277] MUST be used for all internationalized strings sent by the peer.

10. IANA Considerations

This document defines a number of "magic" numbers to be maintained by the IANA. This section explains the criteria 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.

Sections 10.1 through 10.3 are requests for new values already managed by IANA according to [RFC3438].

The remaining sections are for new registries that have been added to the existing L2TP registry and are maintained by IANA accordingly.

10.1. Control Message Attribute Value Pairs (AVPs)

This number space is managed by IANA as per [RFC3438].

A summary of the new AVPs follows:

Control Message Attribute Value Pairs

Attribute Type	Description
58	Extended Vendor ID AVP
59	Message Digest
60	Router ID
61	Assigned Control Connection ID
62	Pseudowire Capabilities List
63	Local Session ID
64	Remote Session ID
65	Assigned Cookie
66	Remote End ID
68	Pseudowire Type
69	L2-Specific Sublayer
70	Data Sequencing
71	Circuit Status
72	Preferred Language
73	Control Message Authentication Nonce
74	Tx Connect Speed
75	Rx Connect Speed

10.2. Message Type AVP Values

This number space is managed by IANA as per [RFC3438]. There is one new message type, defined in Section 3.1, that was allocated for this specification:

Message Type AVP (Attribute Type 0) Values

Control Connection Management

20 (ACK) Explicit Acknowledgement

10.3. Result Code AVP Values

This number space is managed by IANA as per [RFC3438].

New Result Code values for the CDN message are defined in Section 5.4. The following is a summary:

Result Code AVP (Attribute Type 1) Values

General Error Codes

13 - Session not established due to losing tie breaker (L2TPv3).
14 - Session not established due to unsupported PW type (L2TPv3).
15 - Session not established, sequencing required without valid L2-Specific Sublayer (L2TPv3).
16 - Finite state machine error or timeout.

10.4. AVP Header Bits

This is a new registry for IANA to maintain.

Leading Bits of the L2TP AVP Header

There six bits at the beginning of the L2TP AVP header. New bits are assigned via Standards Action [RFC2434].

Bit 0 - Mandatory (M bit)
Bit 1 - Hidden (H bit)
Bit 2 - Reserved
Bit 3 - Reserved
Bit 4 - Reserved
Bit 5 - Reserved

10.5. L2TP Control Message Header Bits

This is a new registry for IANA to maintain.

Leading Bits of the L2TP Control Message Header

There are 12 bits at the beginning of the L2TP Control Message Header. Reserved bits should only be defined by Standard Action [RFC2434].

Bit 0 - Message Type (T bit)
Bit 1 - Length Field is Present (L bit)
Bit 2 - Reserved
Bit 3 - Reserved
Bit 4 - Sequence Numbers Present (S bit)
Bit 5 - Reserved
Bit 6 - Offset Field is Present [RFC2661]
Bit 7 - Priority Bit (P bit) [RFC2661]
Bit 8 - Reserved
Bit 9 - Reserved
Bit 10 - Reserved
Bit 11 - Reserved

10.6. Pseudowire Types

This is a new registry for IANA to maintain, there are no values assigned within this document to maintain.

L2TPv3 Pseudowire Types

The Pseudowire Type (PW Type, see Section 5.4) is a 2-octet value used in the Pseudowire Type AVP and Pseudowire Capabilities List AVP defined in Section 5.4.3. 0 to 32767 are assignable by Expert Review [RFC2434], while 32768 to 65535 are assigned by a First Come First Served policy [RFC2434]. There are no specific pseudowire types assigned within this document. Each pseudowire-specific document must allocate its own PW types from IANA as necessary.

10.7. Circuit Status Bits

This is a new registry for IANA to maintain.

Circuit Status Bits

The Circuit Status field is a 16-bit mask, with the two low order bits assigned. Additional bits may be assigned by IETF Consensus [RFC2434].

Bit 14 - New (N bit)
Bit 15 - Active (A bit)

10.8. Default L2-Specific Sublayer bits

This is a new registry for IANA to maintain.

Default L2-Specific Sublayer Bits

The Default L2-Specific Sublayer contains 8 bits in the low-order portion of the header. Reserved bits may be assigned by IETF Consensus [RFC2434].

Bit 0 - Reserved
Bit 1 - Sequence (S bit)
Bit 2 - Reserved
Bit 3 - Reserved
Bit 4 - Reserved
Bit 5 - Reserved
Bit 6 - Reserved
Bit 7 - Reserved

10.9. L2-Specific Sublayer Type

This is a new registry for IANA to maintain.

L2-Specific Sublayer Type

The L2-Specific Sublayer Type is a 2-octet unsigned integer. Additional values may be assigned by Expert Review [RFC2434].

0 - No L2-Specific Sublayer
1 - Default L2-Specific Sublayer present

10.10. Data Sequencing Level

This is a new registry for IANA to maintain.

Data Sequencing Level

The Data Sequencing Level is a 2-octet unsigned integer
Additional values may be assigned by Expert Review [RFC2434].

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

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and Palter, B., "Layer Two Tunneling Layer Two Tunneling Protocol (L2TP)", RFC 2661, August 1999.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3066] Alvestrand, H., "Tags for the Identification of Languages", BCP 47, RFC 3066, January 2001.

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

[RFC3438] Townsley, W., "Layer Two Tunneling Protocol (L2TP) Internet Assigned Numbers Authority (IANA) Considerations Update", BCP 68, RFC 3438, December 2002.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

11.2. Informative References

[RFC1034] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987.

[RFC1191] Mogul, J. and S. Deering, "Path MTU Discovery", RFC 1191, November 1990.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC1661] Simpson, W., Ed., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[RFC1700] Reynolds, J. and Postel, J., "Assigned Numbers", STD 2, RFC 1700, October 1994.

[RFC1750] Eastlake, D., Crocker, S., and Schiller, J., "Randomness Recommendations for Security", RFC 1750, December 1994.

[RFC1958] Carpenter, B., Ed., "Architectural Principles of the Internet", RFC 1958, June 1996.

[RFC1981] McCann, J., Deering, S., and Mogul, J., "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC2072] Berkowitz, H., "Router Renumbering Guide", RFC 2072, January 1997.

[RFC2104] Krawczyk, H., Bellare, M., and Canetti, R., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2341] Valencia, A., Littlewood, M., and Kolar, T., "Cisco Layer Two Forwarding (Protocol) L2F", RFC 2341, May 1998.

[RFC2401] Kent, S. and Atkinson, R., "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2581] Allman, M., Paxson, V., and Stevens, W., "TCP Congestion Control", RFC 2581, April 1999.

[RFC2637] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and Zorn, G., "Point-to-Point Tunneling Protocol (PPTP)", RFC 2637, July 1999.

[RFC2732] Hinden, R., Carpenter, B., and Masinter, L., "Format for Literal IPv6 Addresses in URL's", RFC 2732, December 1999.

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

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

[RFC3355] Singh, A., Turner, R., Tio, R., and Nanji, S., "Layer Two Tunnelling Protocol (L2TP) Over ATM Adaptation Layer 5 (AAL5)", RFC 3355, August 2002.

[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.

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

12. Acknowledgments

Many of the protocol constructs were originally defined in, and the text of this document began with, RFC 2661, "L2TPv2". RFC 2661 authors are W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn and B. Palter.

The basic concept for L2TP and many of its protocol constructs were adopted from L2F [RFC2341] and PPTP [RFC2637]. Authors of these versions 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" version, which defined the use of L2TP for tunneling of various L2 payload types (initially, Ethernet and Frame Relay).

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.

Some constructs of L2TPv3 were based in part on UTI (Universal Transport Interface), which was originally conceived by Peter Lothberg and Tony Bates.

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

Juha Heinanen provided helpful review in the early stages of this effort.

Jan Vilhuber, Scott Fluhrer, David McGrew, Scott Wainner, Skip Booth and Maria Dos Santos contributed to the Control Message Authentication Mechanism as well as general discussions of security.

James Carlson, Thomas Narten, Maria Dos Santos, Steven Bellovin, Ted Hardie, and Pekka Savola provided very helpful review of the final versions of text.

Russ Housley provided valuable review and comment on security, particularly with respect to the Control Message Authentication mechanism.

Pekka Savola contributed to proper alignment with IPv6 and inspired much of Section 4.1.4 on fragmentation.

Aside of his original influence and co-authorship of RFC 2661, Glen Zorn helped get all of the language and character references straight in this document.

A number of people provided valuable input and effort for RFC 2661, on which this document was based:

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 wrote the first version of the IANA Considerations section.

Dory Leifer made valuable refinements to the protocol definition of L2TP and contributed to the editing of early versions 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 original endpoint authentication mechanism.

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] (this algorithm is also described in [RFC2581]).

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 ACK message 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 an 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
                        <-       ACK
                        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 an ACK message)

                                <-       ACK
                                Nr: 3, Ns: 2
```

```
      (LCCE B's retransmit timer fires)

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

                                          <-       ACK
                                          Nr: 4, Ns: 2
```

Appendix C: Processing Sequence Numbers

The Default L2-Specific Sublayer, defined in Section 4.6, provides a 24-bit field for sequencing of data packets within an L2TP session. L2TP data packets are never retransmitted, so this sequence is used only to detect packet order, duplicate packets, or lost packets.

The 24-bit Sequence Number field of the Default L2-Specific Sublayer contains a packet sequence number for the associated session. Each sequenced data packet that is sent must contain the sequence number, incremented by one, of the previous sequenced packet sent on a given L2TP session. Upon receipt, any packet with a sequence number equal to or greater than the current expected packet (the last received in-order packet plus one) should be considered "new" and accepted. All other packets are considered "old" or "duplicate" and discarded. Note that the 24-bit sequence number space includes zero as a valid sequence number (as such, it may be implemented with a masked 32-bit counter if desired). All new sessions MUST begin sending sequence numbers at zero.

Larger or smaller sequence number fields are possible with L2TP if an alternative format to the Default L2-Specific Sublayer defined in this document is used. While 24 bits may be adequate in a number of circumstances, a larger sequence number space will be less susceptible to sequence number wrapping problems for very high session data rates across long dropout periods. The sequence number processing recommendations below should hold for any size sequence number field.

When detecting whether a packet sequence number is "greater" or "less" than a given sequence number value, wrapping of the sequence number must be considered. This is typically accomplished by keeping a window of sequence numbers beyond the current expected sequence number for determination of whether a packet is "new" or not. The window may be sized based on the link speed and sequence number space and SHOULD be configurable with a default equal to one half the size of the available number space (e.g., 2^(n-1), where n is the number of bits available in the sequence number).

Upon receipt, packets that exactly match the expected sequence number are processed immediately and the next expected sequence number incremented. Packets that fall within the window for new packets may either be processed immediately and the next expected sequence number updated to one plus that received in the new packet, or held for a very short period of time in hopes of receiving the missing packet(s). This "very short period" should be configurable, with a default corresponding to a time lapse that is at least an order of magnitude less than the retransmission timeout periods of higher layer protocols such as TCP.

For typical transient packet mis-orderings, dropping out-of-order packets alone should suffice and generally requires far less resources than actively reordering packets within L2TP. An exception is a case in which a pair of packet fragments are persistently retransmitted and sent out-of-order. For example, if an IP packet has been fragmented into a very small packet followed by a very large packet before being tunneled by L2TP, it is possible (though admittedly wrong) that the two resulting L2TP packets may be consistently mis-ordered by the PSN in transit between L2TP nodes. If sequence numbers were being enforced at the receiving node without any buffering of out-of-order packets, then the fragmented IP packet may never reach its destination. It may be worth noting here that this condition is true for any tunneling mechanism of IP packets that includes sequence number checking on receipt (i.e., GRE [RFC2890]).

Utilization of a Data Sequencing Level (see Section 5.4.3) of 1 (only non-IP data packets require sequencing) allows IP data packets being tunneled by L2TP to not utilize sequence numbers, while utilizing sequence numbers and enforcing packet order for any remaining non-IP data packets. Depending on the requirements of the link layer being tunneled and the network data traversing the data link, this is sufficient in many cases to enforce packet order on frames that require it (such as end-to-end data link control messages), while not on IP packets that are known to be resilient to packet reordering.

If a large number of packets (i.e., more than one new packet window) are dropped due to an extended outage or loss of sequence number state on one side of the connection (perhaps as part of a forwarding plane reset or failover to a standby node), it is possible that a large number of packets will be sent in-order, but be wrongly detected by the peer as out-of-order. This can be generally characterized for a window size, w, sequence number space, s, and number of packets lost in transit between L2TP endpoints, p, as follows:

If s > p > w, then an additional (s - p) packets that were otherwise received in-order, will be incorrectly classified as out-of-order and dropped. Thus, for a sequence number space, s = 128, window size, w = 64, and number of lost packets, p = 70; 128 - 70 = 58 additional packets would be dropped after the outage until the sequence number wrapped back to the current expected next sequence number.

To mitigate this additional packet loss, one MUST inspect the sequence numbers of packets dropped due to being classified as "old" and reset the expected sequence number accordingly. This may be accomplished by counting the number of "old" packets dropped that were in sequence among themselves and, upon reaching a threshold, resetting the next expected sequence number to that seen in the arriving data packets. Packet timestamps may also be used as an indicator to reset the expected sequence number by detecting a period of time over which "old" packets have been received in-sequence. The ideal thresholds will vary depending on link speed, sequence number space, and link tolerance to out-of-order packets, and MUST be configurable.

Editors' Addresses

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

EMail: jedlau@cisco.com

W. Mark Townsley
cisco Systems

EMail: mark@townsley.net

Ignacio Goyret
Lucent Technologies

EMail: igoyret@lucent.com

Full Copyright Statement

Copyright (C) The Internet Society (2005).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights 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; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat 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 on-line IPR repository at http://www.ietf.org/ipr.

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 implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

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