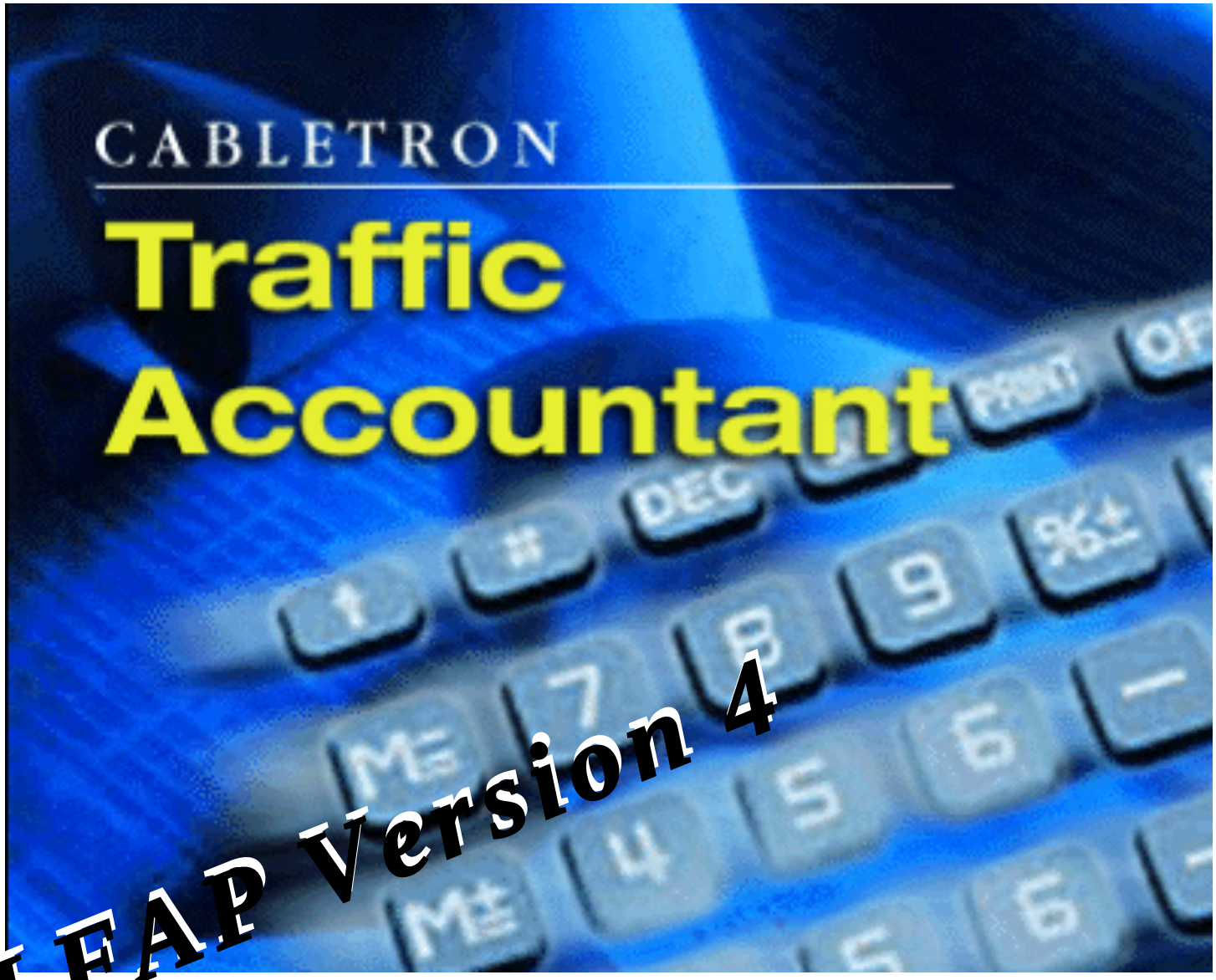


LFAP/FAS API Client Implemenation Guide

CABLETRON SYSTEMS

Notice

Cabletron Systems reserves the right to make changes in specifications and other information contained in this document without prior notice. The reader should in all cases consult Cabletron Systems to determine whether any such changes have been made.

The hardware, firmware, or software described in this manual is subject to change without notice.

IN NO EVENT SHALL CABLETRON SYSTEMS BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS MANUAL OR THE INFORMATION CONTAINED IN IT, EVEN IF CABLETRON SYSTEMS HAS BEEN ADVISED OF, KNOWN, OR SHOULD HAVE KNOWN, THE POSSIBILITY OF SUCH DAMAGES.

Printed in the United States of America.

Order Number: 9033388-E1

Cabletron Systems, Inc.
P.O. Box 5005
Rochester, NH 03866-5005

Virus Disclaimer

Cabletron Systems makes no representations or warranties to the effect that the Licensed Software is virus-free.

Cabletron has tested its software with current virus checking technologies. However, because no anti-virus system is 100% reliable, we strongly caution you to write protect and then verify that the Licensed Software, prior to installing it, is virus-free with an anti-virus system in which you have confidence.

Restricted Rights Notice

(Applicable to licenses to the United States Government only.)

1. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

   Cabletron Systems, Inc., 35 Industrial Way, Rochester, New Hampshire 03866-5005.

2. (a) This computer software is submitted with restricted rights. It may not be used, reproduced, or disclosed by the Government except as provided in paragraph (b) of this Notice or as otherwise expressly stated in the contract.

   (b) This computer software may be:

   (1) Used or copied for use in or with the computer or computers for which it was acquired, including use at any Government installation to which such computer or computers may be transferred;

   (2) Used or copied for use in a backup computer if any computer for which it was acquired is inoperative;

   (3) Reproduced for safekeeping (archives) or backup purposes;

   (4) Modified, adapted, or combined with other computer software, provided that the modified, combined, or adapted portions of the derivative software incorporating restricted computer software are made subject to the same restricted rights;

   (5) Disclosed to and reproduced for use by support service contractors in accordance with subparagraphs (b) (1) through (4) of this clause, provided the Government makes such disclosure or reproduction subject to these restricted rights; and

   (6) Used or copied for use in or transferred to a replacement computer.

   (c) Notwithstanding the foregoing, if this computer software is published copyrighted computer software, it is licensed to the Government, without disclosure prohibitions, with the minimum rights set forth in paragraph (b) of this clause.

   (d) Any other rights or limitations regarding the use, duplication, or disclosure of this computer software are to be expressly stated in, or incorporated in, the contract.

   (e) This Notice shall be marked on any reproduction of this computer software, in whole or in part.

Contents

Preface

Who Should Read This Guide vii
Purpose and layout of this document vii
Using This Manual viii

Chapter 1 Introduction

Connection Control Entity (CCE) 1-2
- Architecture 1-2
- Critical Path Flow 1-2
  - Flow Statistics 1-3
  - Flow Management 1-3
  - Unsolicited Messages from the FAS 1-4

Components 1-4
- List of Components 1-4

Component Specifications 1-6
- System Support Software Entry Points and Specifications 1-6
  - Synchronize 1-6
  - Get Uptime 1-6
  - Get Statistics 1-6
  - Set System Timer 1-7
  - Send Message 1-7
  - Send Vector 1-7
  - Connect 1-7
  - Disconnect 1-8
- LFAP API Entry Points and Specifications 1-8
  - Initialize Flow Information 1-8
  - Init 1-8
  - Update Parameters 1-9
  - Terminate 1-9
  - Process Work 1-10
  - Get Flow ID 1-10
  - Admit Flow 1-10
  - Flow Deleted 1-11
  - FAS Connection Lost 1-11
  - Receive Message 1-11
  - Decode Message Length 1-12
  - Get Last Error 1-12
  - Get API Stats 1-12
  - Get Active Server 1-12
  - Get Connection Status 1-13

Chapter 2 LFAP API

Constants, Structures, and Functions....2-1
Notation....2-1
Constants....2-2
Structures....2-6
Functions....2-10
Connection Control Entity Software....2-10
LFAP API....2-11
Memory Usage....2-16
LFAP API Library....2-16
LFAP API System Memory....2-17
Heap....2-17
Stack....2-17
Per Flow Memory....2-17
Heap....2-17
Stack....2-18

Appendix A LFAP Specification (RFC 2124)

Preface

Welcome to the LFAP/FAS Client Implementation Guide. This book is intended as a guide to implementing the Lightweight Flow Accounting Protocol (LFAP) within the Connection Control Entities of a third-party switch. It includes an overview of LFAP, Cabletron's Flow Administration Server (FAS), and a description of LFAP API functions.

Who Should Read This Guide

This guide is intended for use by third-party switch developers wishing to develop LFAP-compliant devices for use in a FAS evironment. This version of the ***LFAP API Client Implementation Guide*** supports LFAP Version 4.

This book assumes you are familiar with the terms and principles associated with traditional network devices such as hubs, routers, and bridges. It also assumes that you are familiar with the C programming language and the LFAP specification (RFC 2124).

Purpose and layout of this document

This document is the Connection Control Entities (CCE) LFAP implementers design guide. It is intended to allow switch vendors to understand the requirements on the switch to support LFAP. This document defines a common structure under which LFAP can run when operating in a CCE environment. It is realized that not all switches nor potentially any switch may break the problem into the exact match that this document uses. However it is believed that by using a model of how to implement LFAP that is not tied to a hardware architecture the vendor will easily see how to apply it to their environment. With this in mind the document has the following layout:

- General Architecture
- Features supported
- Component identification
- Component Specification
- LFAP API

Using This Manual

This document is divided into two chapters and one appendix, as follows:

- Chapter 1, *Introduction* - This chapter gives an overview of the Light-weight Flow Accounting Protocol (LFAP), Cabletron's LFAP API and Cabletron's Flow Accounting Server (FAS).The LFAP API provides an interface for a switch or other Connection Control Entity (CCE) to connect with a FAS.
- Chapter 2, *LFAP API* - This chapter describes the Light-weight Flow Accounting Protocol (LFAP) API functions and structures. This chapter also describes the memory usage of the LFAP API. The LFAP API provides a public interface to the LFAP Server service. Firmware developers use this interface to interact a Connection Control Entity with a Server.
- Appendix A, *LFAP Specification (RFC 2124)* - This appendix is a copy of the Internet Draft of the LFAP Specification (RFC 2124).

Chapter 1

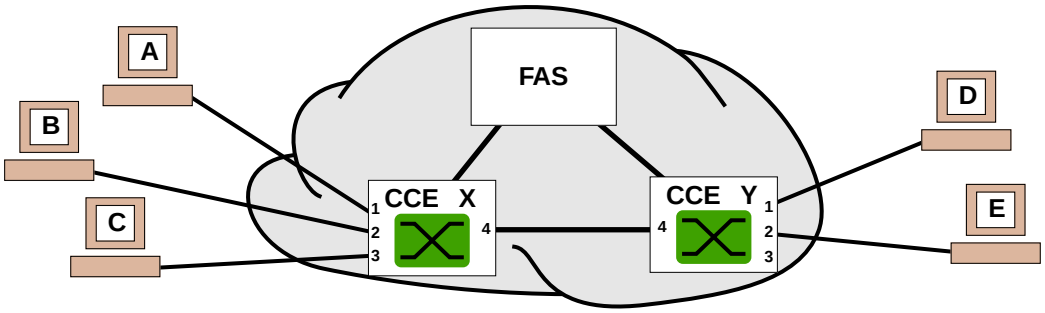
Introduction

This chapter gives an overview of the Light-weight Flow Accounting Protocol (LFAP), Cabletron's LFAP API and Cabletron's Flow Accounting Server (FAS).The LFAP API provides an interface for a switch or other Connection Control Entity (CCE) to connect with a FAS.

The Light-weight Flow Accounting Protocol (LFAP) provides a medium for gathering flow statistics and passing them from an LFAP client in a Connection Control Entity (CCE) to an external Flow Accounting Server (FAS). The CCE, which might be a LAN or ATM switch, must be capable enabling/disabling accounting for ports/interfaces, recognize when a new flow is started and notify the LFAP API. Once a flow has been established, the LFAP API client sends periodic updates on flow statistics (i.e. bytes received, bytes sent) to the server. When the flow is terminated the client informs the server. The flow statistics are then made available to an accounting application that can be used allocate the network costs according to usage, or analyze network traffic. LFAP uses TCP/IP as its transport protocol.

In Figure 1-1, assume that user A opens a connection to user E. CCE X creates a flow and notifies the LFAP API that a new flow has been started. The FAS records a header file for the flow statistics. Subsequent flow updates add to the statistics gathered in the FAS until the flow is terminated. In general the FAS maintains details of flows for billing, capacity planning and other purposes.

Figure 1-1. Flow Accounting



Connection Control Entity (CCE)

The role of the CCE is to admit and track flows and interact with the FAS to record statistics for those flows. The CCE must be capable of configuring the accounting for flows, such that it avoids double billing. This could be as simple as setting accounting on or off at an ingress/egress port, or it could be as complex as defining a set of ACLs for each interface. In the previous example, if accounting is set to capture flow statistics at the User A's ingress (CCE X - port 1), then it should not capture statistics for the same flow at the egress port (CCE X - port 4), the ingress to CCE Y (CCE Y - port 4), or the egress to User E (CCE Y - port 1).

The CCE sits between any two connected entities and either forwards or discards frames based on the policies defined in the CCE. The CCE uses the Light-weight Flow Accounting Protocol (LFAP), RFC 2124, to communicate with the FAS.

The CCE is a logical architectural component and can be provided in a number of forms:

- a stand alone box (e.g. 2 port CCE)
- integrated into a switch (LAN or ATM)
- integrated into a router

For the purposes of this document the CCE will be described as a standalone two port box, but it will be assumed that it can be packaged in many different ways and with as many ports as desired.

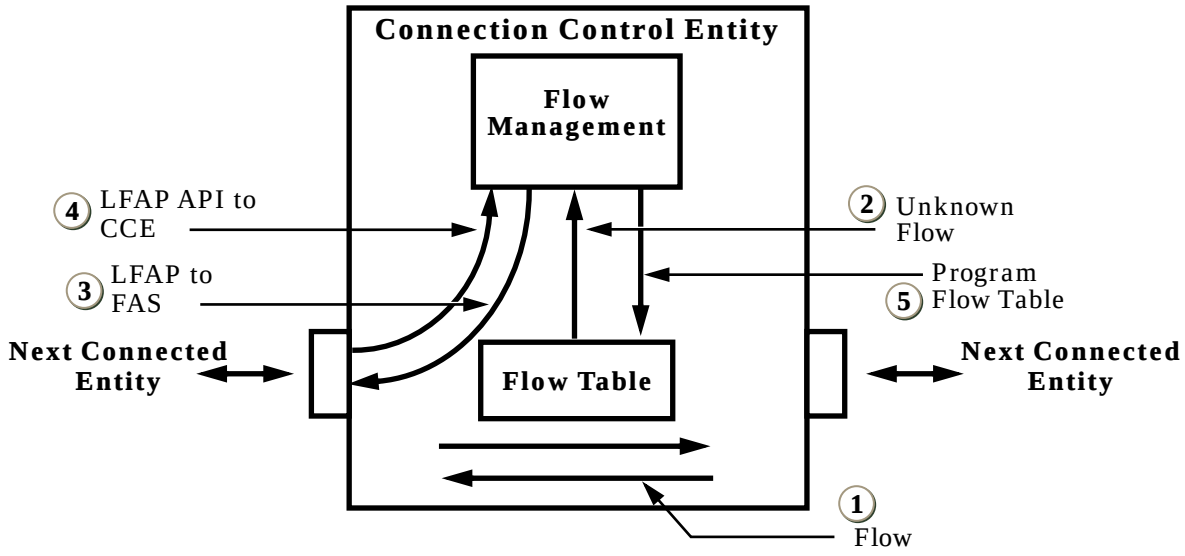
Architecture

Figure 1-2 identifies the logical architecture of a CCE. It consists of a Flow Table where existing flows are mapped and flow management software that maintains the flow table creates and maintains flows and interacts with the FAS to record flow statistics. The features that are required of flow management are described in the next section.

Critical Path Flow

Incoming Frames from either port are examined to determine the flow characteristics. Characteristics that define a flow can include: SA, DA, VPI/VCI, protocol, port number, etc. A flow discriminator (key) is built and used to perform a lookup in the Flow Table. This lookup can have one of three results: FLOW-NOT-FOUND, FILTER or FORWARD. Each of these results causes a different action to be taken. If the result was FORWARD, the frame will be sent to the opposite port. If the result was FILTER the frame is discarded. If the result was FLOW-NOT-FOUND, the frame will be sent to the Flow Management Entity which is considered to be outside the critical path.

Figure 1-2. Connection Control Entity Architecture (with flow setup)



Flow Statistics

The critical path on the CCE is also responsible for keeping statistics on each flow. These statistics are associated with the Flow-Table and may be presented to the Flow-Table as the flow is looked up or maybe added when a frame is forwarded. These include: frame count in each direction and octet-count in each direction. In addition, there may be some global statistics collected for FILTER and FLOW-NOT-FOUND events.

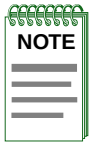
Flow Management

The Flow Management component handles any unidentified flows and manages the Flow-Table for creating and releasing flows.

When a frame arrives at Flow-Management, it either represents a flow to be resolved or a message to Flow-Management itself (e.g. LFAP message from the FAS, SNMP Management Request, etc.).

When an unresolved flow is received and admitted by the CCE, the following actions are taken.

- An LFAP Flow Admission Request (FAR) is sent to the FAS to indicate that a flow has been admitted.
- Program the flow into the Flow Table (FORWARD)
- Forward the Frame to the opposite port from which it was received



Although this section describes communications with the FAS synchronously, this is only done for convenience of description. In reality the requests and responses would be handled asynchronously.

Unsolicited Messages from the FAS

An unsolicited Administrative Request (AR) message, received from the FAS, requests that a Flow Update (FUN) message be returned to update selected flows. The statistics for the requested flows are sent to the FAS in the FUN message(s).

Components

The CCE flow management consists of several components. These components interact with the LFAP server, the CCE operating environment, and the flow/connection data received on CCE ports.

List of Components

The CCE consists of the following components:

- CCE Support Software: This component deals directly with the underlying hardware and CCE system software. It needs to supply the following support for the LFAP API software.

  Connection and flow control – This section recognizes that a flow needs to be established. It requests a Flow Id from the LFAP API software. It then indicates to the LFAP API software that a flow has been started. The CCE must handle establishing the connection and maintaining the relationship of the Flow Id to the connection.

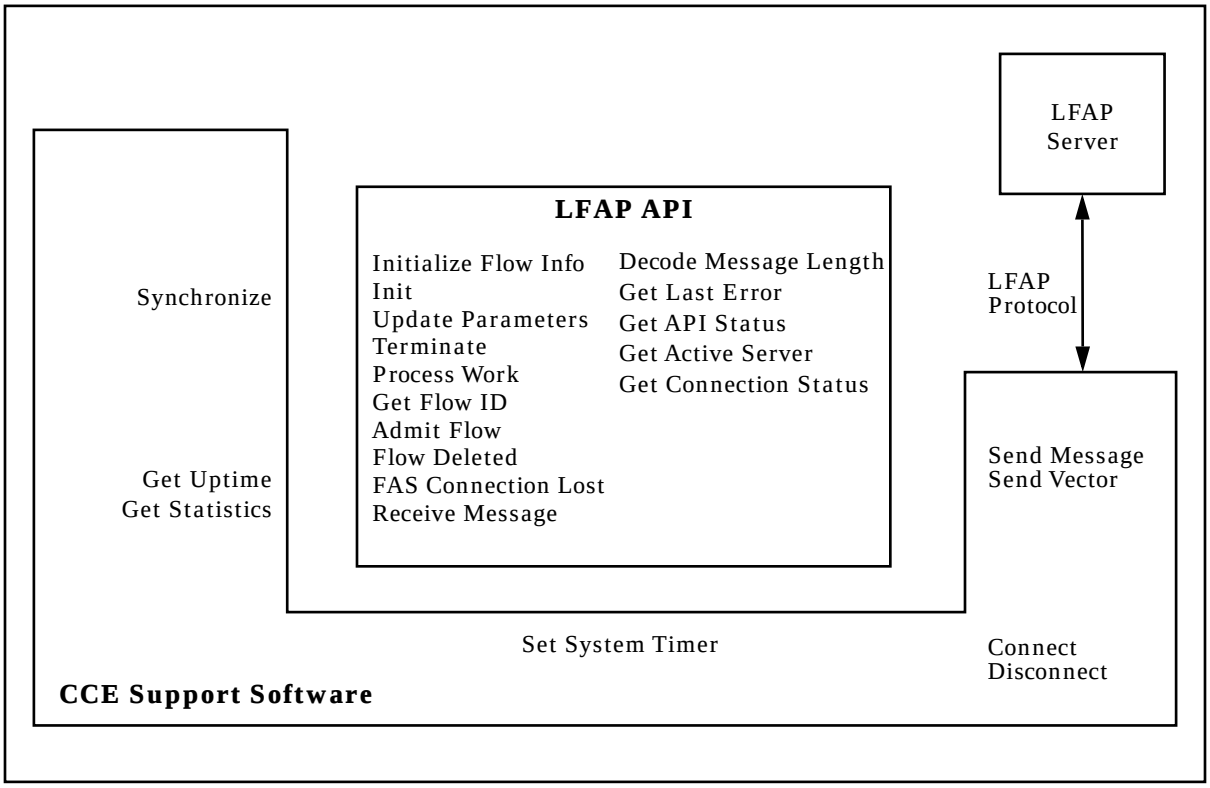
  Flow and system information – This section provides the LFAP API with information about the flow and system uptime.

System timer resources – This section allows the LFAP API to support the timer(s) needed.

Client communication – This section provides the interface to the underlying TCP/IP software.

- LFAP API Software: This component receives flow indications from the system connection and flow control. When the establishment of a flow is indicated it formats the FAR message for the LFAP server and calls upon the client communication software to send the message to the server. If the flow is being removed it formats a FUN message for the LFAP server and calls upon the client communications software to send the message to the server. When it receives messages from the LFAP server it calls upon the appropriate Client system support routines to perform the requested actions.

Figure 1-3. Software Components and Entry Points



Component Specifications

System Support Software Entry Points and Specifications

Synchronize

This entry is called by the LFAP API to allow a non-preemptive system to run other tasks.

- Inputs
  - None
- Outputs
  - None

Get Uptime

This entry returns the CCE uptime value. This is the TimeStamp value defined in RFC 1443.

- Inputs
  - None
- Outputs
  - System Uptime

Get Statistics

This entry point is used to get Statistics on the requested flow

- Inputs
  - Flow ID
  - Empty structure for statistics for flow
- Outputs
  - None

Set System Timer

This entry point is used to set a timer and provide a return entry point when the timer expires.

- Inputs
  - Return Handle
  - CallBack Entrypoint
  - Timer Value
- Outputs
  - None

Send Message

This entry point is used to send a message to the server over the TCP connection.

- Inputs
  - Pointer to the formatted LFAP buffer to be sent
  - Length of the LFAP buffer to be sent
- Outputs
  - Indication of success or failure

Send Vector

This entry point is used to send an array of messages to the server over the TCP connection.

- Inputs
  - Pointer to the an array of LFAP buffers to be sent
  - Number of buffers in the array
  - Total length of the messages in the array
- Outputs
  - Indication of success of failure

Connect

This entry point is used to attempt to make a TCP connection to a server.

- Inputs
  - Server IP address
- Outputs
  - Indication of success or failure

Disconnect

This entry point is used to attempt to break the TCP connection to a server.

- Inputs
  - Server IP address
- Outputs
  - Indication of success or failure

LFAP API Entry Points and Specifications

Initialize Flow Information

This command initializes the flow information structure.

- Inputs
  - Pointer to a flow information structure
- Outputs
  - None

Init

This entry point allows the LFAP API to initialize. This is also where the initial parameters for the CCE can be set. This includes resetting the Flow ID to zero (0) and initializes all the global variables. Init must be called before Process Work is done.

- Inputs
  - Statistics Type (deltas = 0 or cumulative = 1) default = 0
  - Poll Interval (Flow checkpoint timer for all flows that require accounting) default = 1 day in milliseconds
  - Batch Size (Number of FUN's to report on each lesser poll interval. Default = 32
  - Batch interval, in milliseconds, to wait before sending batched messages default = 1 second.
  - Lost connection interval, in milliseconds, defines the timeout value for disconnecting from the FAS. A timeout occurs when there is no response from the FAS within the lost connection interval (default = 10 seconds).
  - Server retry interval, in milliseconds, is the time to wait before going thought the list of servers (default = 1 minute).

  - Max send queue size is the number of messages stored in the send queue before messages are dropped (default = 10,000 messages).
  - Server list is a list of IP addresses to which the CCE can connect.
  - Server list length is the length of the server list.

- Outputs
  - None

Update Parameters

This entry point allows changing the CCE's initial parameters that were set in the Init command.

- Inputs
  - Statistics Type (deltas = 0 or cumulative = 1) default = 0
  - Poll Interval (Flow checkpoint timer for all flows that require accounting) default = 1 day in milliseconds
  - Batch Size (Number of FUN's to report on each lesser poll interval. Default = 32
  - Batch interval, in milliseconds, to wait before sending batched messages default = 1 second.
  - Lost connection interval, in milliseconds, defines the timeout value for disconnecting from the FAS. A timeout occurs when there is no response from the FAS within the lost connection interval (default = 10 seconds).
  - Server retry interval, in milliseconds, is the time to wait before going thought the list of servers (default = 1 minute).
  - Max send queue size is the number of messages stored in the send queue before messages are dropped (default = 10,000 messages).
  - Server list is a list of IP addresses to which the CCE can connect.
  - Server list length is the length of the server list.

- Outputs
  - None

Terminate

This command shuts down the LFAP API, clears memory, and disconnects from the server.

- Inputs
  - None

- Outputs
  - None

Process Work

This is the main work loop for the LFAP API. Every major operation is completed in this function– must be called continually after Init is called. For example, processing timers, sending messages, processing received messages, connecting to the server, etc.

- Inputs
  - None
- Outputs
  - Indication if more work needs to be completed immediately.

Get Flow ID

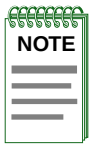
This entry point allows the CCE connection and flow control to request that the LFAP API provide a Flow ID to be used for a recognized flow. A Flow ID of 0 Means no FAS is available.

- Inputs
  - None
- Outputs
  - Next available Flow ID

Admit Flow

This entry point is used to request that the LFAP API initiate a Flow Admit Request (FAR) to the LFAP server. All the information required for the FAR message except for system uptime is provided by the CCE connection and flow control. The flow information structure contains all of the information for the flow, such as, source and destination address and source and destination ports.

- Inputs
  - Flow ID
  - Pointer to a configured flow information structure
- Outputs
  - None



A FAR message is not sent when this function is called, instead it is placed on a queue.

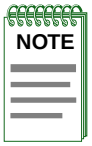
Flow Deleted

This entry point is used by the CCE connection and flow control to indicate to the LFAP API that the connection has been removed and the flow is to be removed.

- Inputs
  - Flow ID
  - Pointer to configured statistics structure

Outputs

  - None



A FUN message is not sent when this function is called, instead it is placed on a queue.

FAS Connection Lost

This entry point is used by the CCE connection and flow control to indicate to the LFAP API that the connection to the FAS has been lost. This will cause sending of messages by the API to be stopped and any new messages generated by flow creations or deletions will be queued.

- Inputs
  - Server IP address
- Outputs
  - None

Receive Message

This entry is used by the client communication software to give the LFAP API the LFAP message received from the LFAP Server.

- Inputs
  - Pointer to the formatted LFAP buffer received
  - Length of the buffer
- Outputs
  - None

Decode Message Length

This entry decodes the length field of the LFAP message header and allows the incoming TCP/IP stream to be divided into individual LFAP messages for processing.

- Inputs
  - Pointer to the beginning of an LFAP message fragment, at least LFAP_HEADER_SIZE in length.
- Outputs
  - Length of message body, excluding the length of the message header.

Get Last Error

This entry returns the last error that occurred in the LFAP API.

- Inputs
  - None
- Outputs
  - Pointer to the error structure containing the error and the time value.

Get API Stats

This entry returns a pointer to the API statistics. The structure has the most recent statistics of the API on a per-server basis.

- Inputs
  - None
- Outputs
  - Pointer to the API Statistics structure.

Get Active Server

This entry returns the IP address of the server where the CCE is currently connected.

- Inputs
  - None
- Outputs
  - IP address to which the CCE is current connected.

Get Connection Status

This entry indicates the current status of the connection to a server.

- Inputs
  - None
- Outputs
  - Indication of connection status. (Refer to Chapter 2 for the details.)

Chapter 2

LFAP API

This chapter describes the Light-weight Flow Accounting Protocol (LFAP) API functions and structures. This chapter also describes the memory usage of the LFAP API. The LFAP API provides a public interface to the LFAP Server service. Firmware developers use this interface to interact a Connection Control Entity with a Server.

Cabletron's Light-weight Flow Accounting Protocol API provides an interface for a switch or other Connection Control Entity (CCE) device to connect with a Flow Accounting Server (FAS). The LFAP API code is written in the C language enabling the code to be used on a multiple of platforms.

This chapter describes the API's functions and their arguments, as well as any structures and constants available to developers. Before using the LFAP API, you should understand the operation of the Flow Accounting Server (FAS) and be familiar with the LFAP protocol.

Constants, Structures, and Functions

The constants, structures, and functions available to a developer are discussed in detail in this section. The two files that need to be included on the CCE side are *LfapFlowMgr.h* (has all of the LFAP API's functions that the CCE will call) and *LfapStruct.h* (has all of the constants and structures the CCE will use). Please refer to the template header file *LfapCCEos_h* when writing the CCE's portion of the code.

Notation

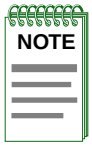
The following notation is used in this section:

() - parenthesis enclose a function's parameters

italics denote an optional parameter

foo (bar, *foobar*) - denotes function 'foo' with mandatory parameter 'bar' and optional parameter 'foobar'

Constants



The following constants are defined in the LfapSysDepend.h file.

```
typedef unsigned char           lfapui8_t
```

The `lfapui8_t` typedef is a classification an 8 bit integer.

```
typedef unsigned short          lfapui16_t
```

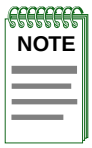
The `lfapui16_t` typedef is a classification of a 16 bit integer.

```
typedef unsigned long           lfapui32_t
```

The `lfapui32_t` typedef is a classification of a 32 bit integer.

```
#define LFAP_SIZEOF_UINT8_T     sizeof(lfapui8_t)
#define LFAP_SIZEOF_UINT16_T    sizeof(lfapui16_t)
#define LFAP_SIZEOF_UINT32_T    sizeof(lfapui32_t)
```

The size of the classifications of integers.



The following constants are defined in the LfapStruct.h file.

```
typedef int (*LFAP_CB_ENTRYPOINT)(void*)
```

The `LFAP_CB_ENTRYPOINT` typedef defines the callback function type used in the LFAP API. The `LFAP_CB_ENTRYPOINT` defines the function that is called back from the timer callbacks. The single argument to the callback function is of type void*, and the return value is of type int.

```
#define CS_DEBUG                1
int lfap_debug
```

If `CS_DEBUG` is defined (as shown above) and `lfap_debug` is not 0, then debugging printouts will be displayed to the terminal upon execution of the program (using `printf` statements). These two variables combined allow for complete flexibility. `CS_DEBUG` is used at compile time allowing printout statements to be compiled into the code. `lfap_debug` is used at run time to dynamically add in printout statements if desired. The debugging printouts can also be compiled in by adding `-DCS_DEBUG` to the CFLAGS compilation flags.

```
#define LFAP_DBG_ERRORS                 1
#define LFAP_DBG_TIMERS                 2
#define LFAP_DBG_WARNINGS               4
#define LFAP_DBG_MESSAGES               8
#define LFAP_DBG_FUNCTIONS              16
#define LFAP_DBG_CONNECTING             32
```

These constants define the level of debugging printouts. As long as `CS_DEBUG` is defined, setting `lfap_debug` with these constants produces the desired printouts.

```
#define LFAP_QUALIFIER_INVALID          0
#define LFAP_AS_NUM_INVALID             0
#define LFAP_TOS_INVALID                255
```

These constants define invalid values for flow qualifiers, AS numbers, and type of service. The appropriate members of the **LFAPFlowInfo** structure are set to these values when they are not being used.

```
#define LFAP_MAX_ADDR_LEN               20
```

The maximum length of an address is 20 characters. The maximum length is set to 20 because the longest address available is an ATM address.

```
#define LFAP_MAX_PROTOCOL_LEN           20
```

The maximum length of the RMON protocol string passed in the **LFAPFlowInfo** structure.

```
#define LFAP_DEFAULT                    0
```

If the default value is desired for any of the tunables passed into `LFAPInit` or `LFAPUpdateParameters`, `LFAP_DEFAULT` can be passed as the parameter.

```
#define LFAP_DEFAULT_POLL_INTERVAL      86400000
```

The poll interval is the time between updates on flows. The default poll interval is set to 24 hours in milliseconds.

```
#define LFAP_DEFAULT_BATCH_SIZE         32
```

The batch size is the number of flow updates that are sent in a single Flow Update Notification (FUN) message. The default batch size is 32 updates per FUN.

```
#define LFAP_DEFAULT_BATCH_INTERVAL     1000
```

The batch interval is the time which indicates how long to wait before sending batched FAR and FUN-Inactive messages. The default batch interval is set to 1 second in milliseconds.

```
#define LFAP_DEFAULT_SRVR_RETRY_INTRVL     60000
```

The server retry interval is the time to wait before going through the list of servers trying to connect to one of them. The default server retry interval is 1 minute in milliseconds.

```
#define LFAP_DEFAULT_LOST_CONN_INTERVAL    10000
```

The lost connection interval is the time to wait for a response from the server before disconnecting and making a connection to the next server in the list. The default lost connection interval is 10 seconds in milliseconds.

```
#define LFAP_DEFAULT_MAX_SEND_QUEUE        10000
```

The maximum send queue size is the maximum number of messages that can be stored in the send queue before messages are dropped. The default maximum send queue size is 10,000 messages.

```
#define LFAP_DELTA_STATS                   0
#define LFAP_TOTAL_STATS                   1
```

These constants define how the LFAP API understands the logging of statistics, either as a total count since the start of the flow, or as a delta amount since the last time statistics were sent up to the server.

```
#define LFAP_NO_STATS_AVAILABLE            0
#define LFAP_BYTES_STATS                   1
#define LFAP_PACKETS_STATS                 2
#define LFAP_BOTH_STATS                    3
```

These constants define the type of statistics that the CCE is returning for a flow if the LFAP API calls the CCE `LFAPGetStatistics` function. It is also passed to the LFAP API `LFAPFlowDeleted` function. The CCE must assign the `type` member of the **LFAPFlowStats** structure to one of these constants. `LFAP_NO_STATS_AVAILABLE` indicates that the CCE could not find statistics for the flow. `LFAP_BYTES_STATS` indicates that only the byte counts are being given for the flow. `LFAP_PACKETS_STATS` indicates that only packet counts are being given for the flow. `LFAP_BOTH_STATS` indicates that both byte and packet counts are being given for the flow.

```
#define LFAP_SEND_FAILURE                  0
#define LFAP_SEND_SUCCESS                  1
```

These constants indicate if a message was sent successfully from the CCE to the server. These constants are returned from the `LFAPSendMessage` or `LFAPSendVector` functions on the CCE.

```
#define LFAP_MAX_MSGS_IN_SEND_VECTOR      16
```

This constant indicates the maximum number of messages that are allowed in the send vector that is sent to the CCE `LFAPSendVector` function.

```
#define LFAP_NOT_TRYING_TO_CONNECT         0
#define LFAP_ALREADY_TRYING_TO_CONNECT    -2
```

These constants are used to indicate if the CCE is already trying to connect to a server. The `LFAP_ALREADY_TRYING_TO_CONNECT` constant could be returned from the LFAP API `LFAPGetConnectionStatus` function.

```
#define LFAP_TCP_CONNECTION_ESTABLISHED    1
#define LFAP_TCP_CONNECTION_FAILED        -1
#define LFAP_TCP_DISCONNECT_SUCCESSD       1
#define LFAP_TCP_DISCONNECT_FAILED        -1
```

These constants indicate if an attempt by the LFAP API to either make or break a connection to the server was a success or a failure. `LFAP_TCP_CONNECTION_FAILED` could be returned from the LFAP API `LFAPGetConnectionStatus` function.

```
#define LFAP_TCP_CONNECTED_NO_ACK          2
```

This constant indicates if a TCP connection has been made from the CCE to the server, but an acknowledgment has not been received yet. This could be returned from the LFAP API `LFAPGetConnectionStatus` function.

```
#define LFAP_CONNECTION_ESTABLISHED        0
#define LFAP_CONNECTION_LOST               1
```

These constants indicate if a TCP connection has been established from the CCE to the server, or if the connection has been lost (and the CCE has not tried to connect to another server yet). These constants could be returned from the LFAP API `LFAPGetConnectionStatus` function.

```
#define LFAP_NO_ERROR_OCCURRED             0
#define LFAP_NO_SERVERS_IN_LIST            1
#define LFAP_NO_SERVERS_AVAILABLE          2
#define LFAP_INCOMPATIBLE_VERSION          3
#define LFAP_RCVD_PARTIAL_MESSAGE          4
#define LFAP_INVALID_FLOW_PREFIX_RCVD      5
#define LFAP_LOST_CONN_INTERVAL_EXPIRED    6
```

These constants indicate the possible error conditions that can occur in the LFAP API. The CCE can get the most recent error and the time it occurred by calling the LFAP API `LFAPGetLastError` function.

```
#define LFAP_WORK_COMPLETED                0
#define LFAP_WORK_INCOMPLETE               1
```

These constants indicate if more work needs to be done immediately, or if the LFAP API has no work to do for now. The LFAP API `LFAPProcessWork` function returns one of these constants.

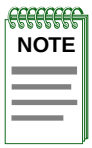
```
#define LFAP_TIMER_INACTIVE                0
#define LFAP_TIMER_WAITING                 1
#define LFAP_TIMER_EXPIRED                 2
#define LFAP_TIMER_PROCESSING              3
```

These constants are used internally to the LFAP API. They indicate the status of each of the timers. `LFAP_TIMER_INACTIVE` indicates that the timer is currently not set. `LFAP_TIMER_WAITING` indicates that the timer is currently set, and waiting. `LFAP_TIMER_EXPIRED` indicates that the timer has just recently expired. `LFAP_TIMER_PROCESSING` indicates that the LFAP API is currently processing the timer.

```
#define LFAP_NOT_REPORTING_FLOWS           0
#define LFAP_REPORT_INDICATED_FLOW         1
```

These constants are used internally to the LFAP API. They indicate if the LFAP API is currently reporting on indicated flows, or not.

Structures



The following structures are described in the LfapStruct.h file.

```
typedef struct lfap_flow_statistics
{
    lfapui16_t      type;
    lfapui32_t      byte_rcvd[2];
    lfapui32_t      byte_sent[2];
    lfapui32_t      packet_rcvd[2];
    lfapui32_t      packet_sent[2];
} LFAPFlowStats;
```

The **LFAPFlowStats** structure is passed to the `LFAPGetStatistics` function on the CCE. It is also passed to the LFAP API `LFAPFlowDeleted` function. The CCE has to fill in the structure for the flow. All of the members (other than the `type` member) of the structure are 2 element arrays, with the name

of the array explaining their intended content. For each array of the **LFAPFlowStats** structure, the first element (e.g. `byte_rcvd[0]`) is the high value, and the second element (e.g. `byte_rcvd[1]`) is the low value. The `type` member must be set to the appropriate constant defined in the section titled *Constants*, on Page 2-2, (either `LFAP_NO_STATS_AVAILABLE`, `LFAP_BYTES_STATS`, `LFAP_PACKETS_STATS`, or `LFAP_BOTH_STATS`).

```
typedef struct lfap_flow_information
{
    char            src_addr[LFAP_MAX_ADDR_SIZE];
    lfapui16_t      src_fam;
    lfapui16_t      src_len;
    char            dest_addr[LFAP_MAX_ADDR_SIZE];
    lfapui16_t      dest_fam;
    lfapui16_t      dest_len;
    lfapui16_t      src_port;
    lfapui16_t      src_port_type;
    char            protocol[LFAP_MAX_PROTOCOL_LEN];
    lfapui16_t      type_of_service;
    void*           client_data;
    unsigned int    client_data_len;
    char            src_cce_addr[LFAP_MAX_ADDR_LEN];
    lfapui16_t      src_cce_fam;
    lfapui16_t      src_cce_len;
    lfapui16_t      src_as;
    lfapui16_t      dest_as;
    lfapui16_t      ingress_port;
    lfapui16_t      egress_port;
    lfapui16_t      cxn_priority;
    lfapui16_t      checkpoint'
} LFAPFlowInfo;
```

The **LFAPFlowInfo** structure is passed from the CCE to the LFAP API when `LFAPAdmitFlow` is called. The CCE has to fill in the structure with information concerning the flow. The source address (`src_addr`) along with its related information (family: `src_fam` and length: `src_len`) and the destination address (`dest_addr`) along with its related information are required to be filled in by the CCE. The `src_port`, `src_port_type`, `protocol`, `type_of_service`, `client_data` (and len), `src_cce_addr` (and related information), `src_as`, `dest_as`, `ingress_port`, `egress_port`, `cxn_priority`, and checkpoint are optional when `LFAPAdmitFlow` is called. Notice that the character buffers (`src_addr`, `dest_addr`, and `src_cce_addr`) are actually arrays that need to be filled in appropriately. Each member of one of the address arrays is a field of the address (i.e. an IP address would use 4 members of the character array, a MAC address would use 6, and an ATM address would use all `LFAP_MAX_ADDR_SIZE` members). For example, if the address is of type IP, then each char will be a value between 1 and 255. The family type of the addresses are specified in RFC 1700.

```
typedef struct lfap_error
{
    lfapui16_t      error;
    lfapui32_t      time;
} LFAPError;
```

The **LFAPError** structure is used to indicate to the CCE what was the most recent error that occurred in the LFAP API, and what time the error occurred. The CCE can call the LFAP API `LFAPGetLastError` function to retrieve the most recent error. The time is set when an error occurs by calling the CCE `LFAPGetUptime` function. The possible values of `error` are defined in the section titled, *Constants*, on Page 2-2.

```
typedef struct lfap_api_statistics
{
    lfapui32_t      conn_time;
    lfapui32_t      up_time;
    lfapui16_t      num_servers;
    lfapui32_t      this_server;
    lfapui16_t      this_server_in_list;
    lfapui32_t      retry_conn_success
    lfapui32_t      retry_conn_failure
    lfapui32_t      total_bytes_sent;
    lfapui32_t      total_bytes_rcvd;
    lfapui32_t      msgs_sent_success;
    lfapui32_t      msgs_sent_failure;
    lfapui32_t      msgs_rcvd_corrupt;
    lfapui32_t      msgs_rcvd_error;
    lfapui32_t      msgs_rcvd_success;
    lfapui32_t      msgs_send_buf;
    lfapui32_t      msgs_rcvd_buf;
    lfapui32_t      msg_vr_sent;
    lfapui32_t      msg_vra_rcvd;
    lfapui32_t      msg_far_sent;
    lfapui32_t      msg_fun_sent;
    lfapui32_t      msg_ar_sent;
    lfapui32_t      msg_ar_rcvd;
    lfapui32_t      msg_ara_sent;
    lfapui32_t      msg_ara_rcvd;
    lfapui32_t      msg_unk_rcvd;
    lfapui32_t      dropped_send_msgs;
    lfapui32_t      dropped_send_msgs_conn;
    lfapui32_t      dropped_vr_msgs;
    lfapui32_t      dropped_ar_msgs;
    lfapui32_t      dropped_ara_msgs;
    lfapui32_t      dropped_far_msgs;
    lfapui32_t      dropped_fun_i_msgs;
```

```
    lfapui32_t      dropped_fun_msgs;
    lfapui32_t      dropped_rcvd_msgs;
    lfapui32_t      lost_bytes_sent_high;
    lfapui32_t      lost_bytes_sent_low;
    lfapui32_t      lost_packets_sent_high;
    lfapui32_t      lost_packets_sent_low;
    lfapui32_t      lost_bytes_rcvd_high;
    lfapui32_t      lost_bytes_rcvd_low;
    lfapui32_t      lost_packets_rcvd_high;
    lfapui32_t      lost_packets_rcvd_low;
    lfapui32_t      flow_setups;
    lfapui32_t      flow_teardowns;
    lfapui32_t      active_flows;
    lfapui32_t      peak_msgs_send_buf;
    lfapui32_t      peak_active_flows;
} LFAPApiStats;
```

The **LFAPApiStats** structure is used to store the various statistics that relate to the LFAP API. The CCE can call the LFAP API `LFAPGetApiStats` function to retrieve the most recent set of statistics.

```
#define SIZEOF_LFAP_API_STATS  sizeof(LFAPApiStats)
```

The size (in bytes) of the **LFAPApiStats** structure.

```
typedef struct lfap_msg_holder_struct {
    void*         buf;
    int           len;
} LFAPMsgHolder;
```

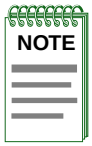
The **LFAPMsgHolder** structure is used to hold a message buffer and its length. The CCE `LFAPSendVector` function takes an array of these structures to send the data.

```
#define SIZEOF_LFAP_MSG_HOLDER sizeof(LFAPMsgHolder)
```

The size (in bytes) of the **LFAPMsgHolder** structure.

Functions

Connection Control Entity Software



The following functions are described in the LfapCCEos_h template header file.

`lfapui32_t LFAPGetUptime (void)`

This function is called by the LFAP API to get the CCE uptime value. The uptime returned is the length of type SNMPv2 TimeStamp (RFC 1443), therefore it is the length of time in hundredths of seconds that the CCE has been operational.

```
void LFAPGetStatistics (lfapui32_t flow_id,
                        LFAPFlowStats* flow_stats)
```

This function is called by the LFAP API to get the statistics on a flow with identifier `flow_id`. A pointer to a zero initialized **LFAPFlowStats** structure is passed into the `LFAPGetStatistics` function to be filled in by the CCE. A reminder that the `type` member of the **LFAPFlowStats** structure must be filled in as well (refer to the section titled *Structures*, on Page 2-6). Note that this function should be autonomous (i.e. no contact switches should occur during the call of this function if the CCE has a threaded operating system).

`void LFAPSynchronize (void)`

This function is called by the LFAP API to allow a non-preemptive system to run other tasks. The CCE should complete whatever work it wishes to complete, then return from this function. Returning from this function will allow the LFAP API to continue with its work. This function should (at a minimum) allow a context switch to occur.

```
void LFAPSetSystemTimer (const LFAP_CB_ENTRYPOINT lfap_cb_entry,
                         lfapui32_t return_handle,
                         lfapui32_t time_val)
```

This function is called by the LFAP API to set a timer for `time_val` milliseconds. `lfap_cb_entry` is the callback entry point that is called with the argument `return_handle` upon completion of the timer.

`int LFAPSendMessage (void* buf, int buf_len)`

This function is called by the LFAP API to send the `buf` to the server. `buf` has encoded in it one of the six different LFAP messages. `buf_len` is the length of the LFAP message, which includes the length of the header of the buffer. `LFAP_SEND_SUCCESS` is returned if successful, `LFAP_SEND_FAILURE` if not.

`int LFAPSendVector (LFAPMsgHolder* vec, int vec_len, int total_len)`

This function is called by the LFAP API to send the messages in the **LFAPMsgHolder** array `vec` to the server. `vec_len` is the number of messages in the array, while `total_len` is the total number of bytes of the messages in the array. `LFAP_SEND_SUCCESS` is returned if successful, `LFAP_SEND_FAILURE` if not.

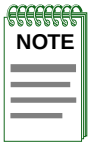
`int LFAPConnect (lfapui32_t server_ip)`

This function is called by the LFAP API to attempt to make a TCP connection to the server with an IP address of `server_ip`. `server_ip` is the unsigned long form of the server's IP address. The server is listening on the LFAP_PORT (port 3145). `LFAP_TCP_CONNECTION_ESTABLISHED` is returned if successful, `LFAP_TCP_CONNECTION_FAILED` is returned if the connection was not able to be completed.

`int LFAPDisconnect (lfapui32_t server_ip)`

This function is called by the LFAP API to attempt to break the TCP connection with the server with an IP address of `server_ip`. `server_ip` is the unsigned long form of the server's IP address. `LFAP_TCP_DISCONNECT_SUCCESS` is returned if successful, `LFAP_TCP_DISCONNECT_FAILED` is returned if the connection was not able to be broken.

LFAP API



The following functions are described in the LfapFlowMgr.h file.

`void initializeLfapFlowInfo (LFAPFlowInfo* flow_info)`

This function is called by the CCE to initialize all of the members of the **LFAPFlowInfo** structure to 0 (or invalid constants). This function is useful to the CCE because the CCE uses the **LFAPFlowInfo** structure when calling the LFAP API `LFAPAdmitFlow` function.

```
int LFAPInit (lfapui16_t stats_type,
              lfapui32_t poll_interval,
              lfapui16_t batch_size,
              lfapui32_t batch_interval,
              lfapui32_t lost_conn_interval,
              lfapui32_t server_retry_interval,
              lfapui32_t max_send_queue_size,
              lfapui32_t* server_list,
              lfapui16_t server_list_len)
```

This function must be called by the CCE to initialize the LFAP API. `stats_type` indicates to the LFAP API what type of statistics the CCE maintains for flows (`LFAP_DELTA_STATS` for delta, `LFAP_TOTAL_STATS` for total). The default is delta statistics. `poll_interval` is a time value in milliseconds which indicates how often statistics for flows are reported to the server. The default poll interval is 24 hours. `batch_size` is the number of flow update notifications that are sent in a single FUN message. The default batch size is 32 updates in one message. `batch_interval` is a time value in milliseconds which indicates how long to wait before sending the batched FAR and FUN-Inactive messages. A group of messages is sent up to the server if the number of messages grouped is equal to the batch size, or if this timer expires. The default `batch_interval` is 1 second. `lost_conn_interval` is a time value in milliseconds which indicates how long the LFAP API will wait for a response from the server after sending a query. If a response is not heard within this time value, the LFAP API assumes the connection to the server is lost. The default lost connection interval is 10 seconds. `server_retry_interval` is a time value in milliseconds which indicates how long to wait before going through the list of servers trying to connect to one of them. The default server retry interval is 1 minute. `max_send_queue_size` is the maximum number of messages that will be stored in the send queue before messages are dropped. The default maximum send queue size is 10,000 messages. `servier_list` is an array of server IP addresses (in unsigned long form) that the CCE can connect to. `servier_list_len` is the length of the server list array. This function must be called before `LFAPProcessWork` is called.

```
void LFAPUpdateParameters (lfapui16_t stats_type,
                           lfapui32_t poll_interval,
                           lfapui16_t batch_size,
                           lfapui32_t batch_interval,
                           lfapui32_t lost_conn_interval,
                           lfapui32_t server_retry_interval,
                           lfapui32_t max_send_queue_size,
                           lfapui32_t* servier_list,
                           lfapui16_t servier_list_len)
```

This function is called by the CCE to change any of the original parameters that were specified with the call to the `LFAPInit` function. `stats_type` indicates to the LFAP API what type of statistics the CCE maintains for flows.

`poll_interval` is a time value in milliseconds which indicates how often statistics for flows are reported to the server. `batch_size` is the number of flow update notifications that are sent in a single FUN message. `batch_interval` is a time value in milliseconds which indicates how long to wait before sending the batched FAR and FUN-Inactive messages. `lost_conn_interval` is a time value in milliseconds which indicates how long the LFAP API will wait for a response from the server after sending a query. If a response is not heard within this time value, the LFAP API assumes the connection to the server is lost. `server_retry_interval` is a time value in milliseconds which indicates how long to wait before going through the list of servers trying to connect to one of them. `max_send_queue_size` is the maximum number of messages that will be stored in the send queue before messages are dropped. `service_list` is an array of server IP addresses (in unsigned long form) that the CCE can connect to. `service_list_len` is the length of the server list array. If the CCE calls this function but does not specify a parameter (e.g. one or more of the parameters is `LFAP_DEFAULT`) then the parameter will be set to its default value (excluding the `server_list` and the `server_list_len`, these will stay their original value if `LFAP_DEFAULT` is passed in for either).

`void LFAPTerminate (void)`

This function shuts down the LFAP API. The LFAP API cleans up its memory, and disconnects from the server (by calling the CCE `LFAPDisconnect` function). After this function is called, the CCE should not return any timer callbacks, nor should it call `LFAPProcessWork`, `LFAPReceiveMessage`, or `LFAPConnectionLost`.

`lfapui16_t LFAPProcessWork (void)`

This function is called by the CCE to allow the LFAP API to do work. Every major operation is completed in this function, in an incremental fashion. This function will not always complete all of the work the LFAP API has to do, it will do an appropriate amount of work before returning. The work that is completed in this function is as follows: processing timers, sending messages, processing received messages, flow updates, connecting to a server, and failing over to a new server. If more work needs to be done immediately, `LFAP_WORK_INCOMPLETE` is returned, otherwise `LFAP_WORK_COMPLETED` is returned. Note that this function must be called continually, even if `LFAP_WORK_COMPLETED` is returned. This function should be called in an infinite loop, as long as the LFAP API has been initialized (with a call to `LFAPInit`). This function should be called up until `LFAPTerminate` is called to shut down the LFAP API.

`lfapui32_t LFAPGetFlowId (void)`

This function is called by the CCE to request that the LFAP API provide a Flow ID to be used for a recognized flow. A return value of 0 indicates that no server is available (invalid Flow ID). This function must be called before `LFAPAdmitFlow` is called.

`void LFAPAdmitFlow (lfapui32_t flow_id, LFAPFlowInfo* flow_info)`

This function is called by the CCE to request that the LFAP API initiate a Flow Admission Request (FAR) to the server. The `flow_id` must be a unique identifier having been obtained by calling the `LFAPGetFlowId` function prior to calling `LFAPAdmitFlow`. `flow_info` is a pointer to a structure that contains all of the relevant information for the definition of the flow (the **LFAPFlowInfo** structure is defined in the Structures section). The following goes through the elements of the **LFAPFlowInfo** structure. The source address (`src_addr`) along with its related information (family: `src_fam` and length: `src_len`) and the destination address (`dest_addr`) along with its related information is required to be defined by the CCE. The `src_port`, `src_port_type`, `protocol`, `type_of_service`, `client_data`, `client_data_len`, `src_cce_addr` and its related information `src_cce_fam` and `src_cce_len`, `src_as`, `dest_as`, `ingress_port`, `egress_port`, `cxn_priority`, and `checkpoint` are all optional. The CCE is responsible for managing the memory used by the **LFAPFlowInfo** structure.

`void LFAPFlowDeleted (lfapui32_t flow_id, LFAPFlowStats* flow_stats)`

This function is called by the CCE to indicate to the LFAP API that the connection has been removed and the flow is to be removed. `flow_id` is the unique identifier for the flow. `flow_stats` is a pointer to a **LFAPFlowStats** structure that has already been filled in with the final statistics for the flow (required). The CCE is responsible for managing the memory used by the **LFAPFlowStats** structure.

`void LFAPConnectionLost (lfapui32_t server_ip)`

This function is called by the CCE to indicate to the LFAP API that the connection to the server has been lost. `server_ip` is the unsigned long form of the IP address of the server with whom connection was lost. This function will return immediately, but the LFAP API now knows it has lost connection to the server. The `LFAPProcessWork` function will do the work of trying to connect to a server. The LFAP API will immediately try to reconnect to the next server in it's list of servers (calling the CCE `LFAPConnect` function). The LFAP API will continually try to connect to a server (rotating through the list of servers) until the list has been gone through once completely. If none of the servers in the server list can be contacted, then a retry timer is set. Once the timer expires, each of the servers in the list are attempted to be contacted again. If this also fails, then another timer is set, and the retry process begins again. If the CCE wishes to not contact any server, then the CCE can call the LFAP API `LFAPTerminate` function, and not return the timer that was set. LFAP will then be completely disabled.

`void LFAPReceiveMessage (void* buf, int buf_len)`

This function is called by the client communication software in the CCE to give the LFAP API the LFAP message received from the server. `buf` is a pointer to the complete LFAP message (including the header) and `buf_len` is the length of the complete LFAP message (including the length of the header). The client communication software in the CCE needs to do very little when a message is received before calling this function. When a message is received, the client communication software has to read at least `LFAP_HEADER_SIZE` bytes (8 bytes) of the message so that the header is read. The `LFAPDecodeMsgLen` function is called with a pointer to the buffer read as the parameter. The length of the message body (does not include the length of the LFAP message header) in bytes is returned by the function. The LFAP message must be read (now that the length is known) and a pointer to the complete message (which includes the header) is passed to this function, along with the length of the complete LFAP message (includes the length of the header). The message is not processed in this function; it is placed in a receive queue to be processed by `LFAPProcessWork`.

`int LFAPDecodeMsgLen (void* buf)`

This function decodes the length field of an LFAP message header. This enables an incoming TCP/IP stream to be divided into individual LFAP messages for processing. `buf` should point to the beginning of an LFAP message fragment that is at least `LFAP_HEADER_SIZE` bytes (8 bytes) in length. This function returns the length of the message body in bytes. This does not include the length of the LFAP message header (which is `LFAP_HEADER_SIZE` bytes).

`LFAPError* LFAPGetLastError (void)`

This function returns a pointer to an **LFAPError** structure which indicates the most recent error that has occurred and the time it occurred in the LFAP API. If no error has occurred, then the error value is `LFAP_NO_ERROR_OCCURRED`, and the time is 0. When an error occurs, the time is set by calling the CCE `LFAPGetUptime` function. The possible errors are `LFAP_NO_SERVERS_IN_LIST`, `LFAP_NO_SERVERS_AVAILABLE`, `LFAP_INCOMPATIBLE_VERSION`, `LFAP_RCVD_PARTIAL_MESSAGE`, or `LFAP_INVALID_FLOW_PREFIX_RCVD`.

`const LFAPApiStats* LFAPGetApiStats (void)`

This function returns a const pointer to a **LFAPApiStats** structure (refer to the section titled *Structures*, on Page 2-6 for a description) which has the most recent statistics of the LFAP API. If the LFAP API has not been initialized yet (`LFAPInit` has not been called) or if the LFAP API has been terminated (`LFAPTerminate` has been called), then this function will return 0.

`lfapui32_t LFAPGetActiveServer (void)`

This function returns the unsigned long format of the IP address of the server that the CCE is connected to. If the CCE is not connected to a server, 0 is returned.

`int LFAPGetConnectionStatus (void)`

This function returns an integer that indicates the current status of the connection to a server. If the CCE is trying to connect to a server, `LFAP_ALREADY_TRYING_TO_CONNECT` is returned. If there is a TCP connection to the server, but no acknowledgment has been received, `LFAP_TCP_CONNECTED_NO_ACK` is returned. If the CCE is connected and an acknowledgment has been received from the server, `LFAP_CONNECTION_ESTABLISHED` is returned. If the CCE just lost connection to the server, `LFAP_CONNECTION_LOST` is returned. Finally, if the CCE just tried to connect to a server, and the connection failed, `LFAP_TCP_CONNECTION_FAILED` is returned.

Memory Usage

The LFAP API was designed so that it would use very little of the memory available on the CCE. The LFAP API code is constantly being examined to determine how it can use even less memory in the future.

LFAP API Library

The LFAP library, which consists of the LFAP protocol and the LFAP API, is approximately 700 kilobytes with debugging information compiled in, and approximately 575 kilobytes without debugging information compiled in. Currently the LFAP API is implemented in Cabletron's SSR 2000, 8000, and 8600 product line. The library surrounding the API, which includes timers, TCP code, threading, debugging, and SNMP is approximately 1 megabyte.

LFAP API System Memory

Heap

The LFAP API uses heap memory in large amounts in two areas. The statistics that are kept on a per server basis are stored on the heap using the **LFAPApiStats** structure. For each server that was ever known, 186 bytes are used on the heap.

The second area is significantly larger. The LFAP API has both send and receive message queues. The receive queue is insignificant, as the server sends very few messages to the client. The send message queue has a tunable maximum size, which is set so that either no messages are dropped, or so that heap memory is not exceeded. The send message queue will grow depending on the flow setup and teardown rates, the TCP throughput rate to the server, and the load on the server. The default maximum send queue size is 10,000 messages. All LFAP messages are stored on the heap. Each flow setup creates a FAR message, which is between 100 and 150 bytes per message, depending on which optional parameters are used. Each flow teardown creates a FUN inactive message, which is less than 100 bytes per message. There are also periodic updates for long lasting flows, which are packed into one message (32 updates in one message is the default) which are approximately 2,000 bytes per message. Therefore, if the maximum send queue size is at 10,000 and if the send queue reaches its maximum, 1 to 5 megabytes of memory could be used. The number of messages in the send queue can be monitored by observing the `msgs_send_buf` and `peak_msgs_send_buf` in the **LFAPApiStats** structure. The memory for each message is freed as soon as the message is sent.

Stack

The LFAP API has approximately 1 kilobyte of memory on the stack at all times.

Per Flow Memory

The LFAP API also uses some additional memory for each new flow created.

Heap

For each flow, the LFAP API uses 8 bytes of memory from the heap.

Stack

The LFAP API does not use any memory from the stack on a per flow basis.

Appendix A

LFAP Specification (RFC 2124)

This appendix is a copy of the Internet Draft of the LFAP Specification (RFC 2124).

Network Working Group
INTERNET-DRAFTPaul Amsden

Paul Calato
Dana Cook
Todd Crowley
Cabletron Systems Inc.

Light-weight Flow Admission Protocol Specification
Version 4.0
<draft-rfced-info-calato-01.txt>

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ``work in progress."

To learn the current status of any Internet-Draft, please check the ``1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast) or ftp.isi.edu (US West Coast).

Abstract

Light-weight Flow Accounting Protocol, LFAP, allows an external Flow Accounting Service (FAS) to record flow accounting information from the switch, allowing flexible Flow Accounting Services to be deployed by a vendor or customer without changes to, or undue burden on, the switch.

Specifically, this document specifies the protocol between the switch Connection Control Entity (CCE) and the external FAS. Using LFAP, a Flow Accounting Service can maintain details of current or historical flows for billing, capacity planning and other purposes.

Table of Contents

1.0 Introduction 3
2.0 Message Flows 3
3.0 Message Contents and Format 4
4.0 Information Elements (IE's) Formats 5
4.1 Flow ID IE 5
4.2 Source Address IE 5
4.3 Destination Address IE 6
4.4 Flow Qualifier IEs 6
4.5 Time IE 7
4.6 Type of Service IE 7
4.7 Source CCE Address IE 7
4.8 Client Data IE 7
4.9 Command Code IE 8
4.10 FAS IP Address IE 8
4.11 Failure Code IE 9
4.12 Flow Identifier Prefix IE 9
4.13 Flow State IE 9
4.14 Byte Count IE 10
4.15 Packet Count IE 10
4.16 Protocol Identifier IE 11
4.17 Source Port IE 11
4.18 Multiple Record IE 12
4.19 Source and Destination AS IE 12
4.20 Ingress Port IE 13
4.21 Egress Port IE 13
4.22 4Numeric List of IE's 13
5.0 Individual Message contents 14
5.1 Flow Accounting Request (FAR) Message 14
5.2 Flow Update Notification (FUN) Message 15
5.3 Administrative Request (AR) Message 16
5.4 Administrative Request Acknowledge (ARA) Message 16
5.5 Version Request (VR) Message 16
5.6 Version Request Acknowledge (VRA) Message 16
6.0 Session Establishment 17
6.1 Protocol Negotiation 17
6.2 Connection Establishment 17
7.0 Error Handling 17
7.1 VRA Related Error Handling 18
7.2 ARA Related Error Handling 18
8.0 Security Considerations 18
9.0 Author's Addresses 18
10.0 References 18

1.0 Introduction

Light-weight Flow Accounting Protocol, LFAP, allows an external Flow Accounting Service (FAS) to record flow accounting information from the switch, allowing flexible Flow Accounting Services to be deployed by a vendor or customer without changes to, or undue burden on, the switch. It provides a means for switches to send accounting information in a fault tolerant client server architecture.

Specifically, this document specifies the protocol between the switch Connection Control Entity (CCE) and the external FAS. Using LFAP, a Flow Accounting Service can maintain details of current or historical flows for billing, capacity planning and other purposes.

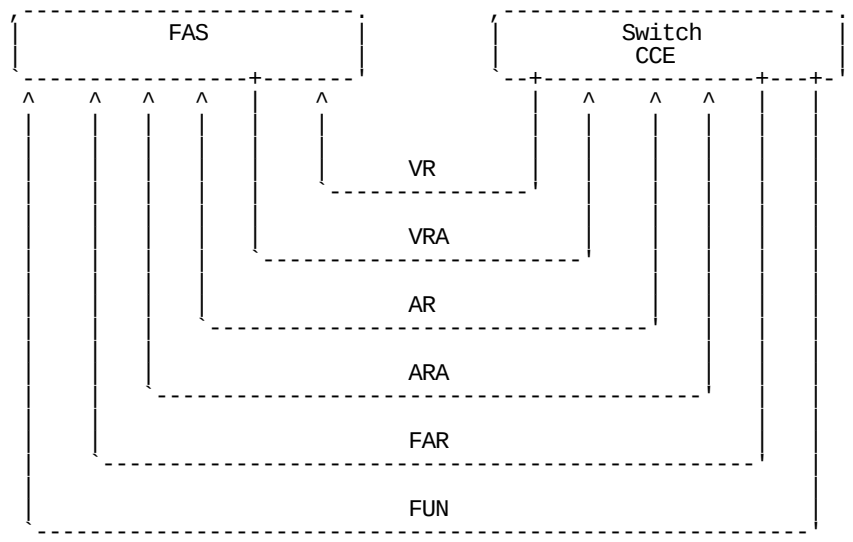
A significant advantage of this protocol is that it provides an efficient manner in which to move flow accounting information from the switch to an accounting application. LFAP uses a push rather than a pull architecture, which enables the switch to send data when necessary and to pack the data into dense messages.

This document describes the message flow between switch CCE and FAS, the messages used and error handling that applies. This constitutes the LFAP interface definition.

2.0 Message Flows

Initiating message flows between CCE and FAS entities always originate at the switch. Therefore, the switch is the point at which connectivity is originated. The CCE must have IP reachability using some approach described elsewhere (e.g. [1577] or [LANE]) and an IP address for the FAS must be preconfigured at the switch CCE. The CCE establishes TCP connectivity using the registered port number 3145 keyword csi-lfap. When the TCP port is opened by the CCE it sends a VR to request a connection using a specific protocol version. The FAS responds with a VRA and a status of success if the version is supported. Once the protocol version is established, the FAS sends an AR with permission for the conversation to continue (i.e., the FAS performs a security check).

As shown below, Flow Accounting Request (FAR) messages are sent by a switch's Call Control Entity (CCE) to the Flow Accounting Service (FAS). These messages are sent when the switch sets up a flow. The message contains specific information relating to the flow, such as flow identifier, source/destination and qualifying information about the flow.



Periodically during the course of maintaining the flow and when a change in flow state occurs, the switch CCE sends a Flow Update Notification (FUN) message to the FAS. The FUN message contains accounting (i.e. bytes received and sent) and other information on a flow.

Either the CCE or the FAS may initiate a Administrative Request (AR). The CCE can use it to get a Flow Identifier Prefix. The FAS can use it to request FUN messages be returned on some set of flows.

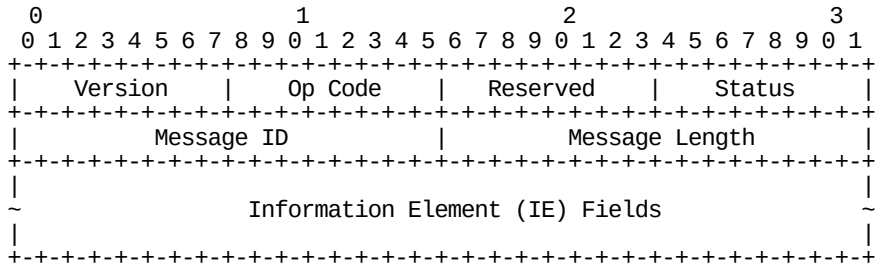
The requested entity (FAS or CCE) replies with an Administrative Request Acknowledge (ARA). The FAS uses the ARA to return the requested Flow Prefix. The CCE uses the ARA to return any Flow Identifiers that were in error on the AR.

3.0 Message Contents and Format

LFAP defines six messages: "Flow Accounting Request", "Flow Update Notification", "Administrative Request", "Administrative Request Acknowledge", "Version Request", "Version Request Acknowledge" (FAR, FUN, AR, ARA, VR, VRA respectively).

FAR messages are sent by a switch call control entity (CCE) to the Flow Accounting Service (FAS). FUN messages originate at switches and AR messages are sent by either the Entity (FAS or CCE) and are acknowledged by the ARA messages.

FSA messages from the FAS.



The general message format for all LFAP messages is as shown above. This is version 4 and Op Codes are as follows:

VR - 1
VRA - 2
FAR - 3
FUN - 4
AR - 5
ARA - 6

The Status field serves as a Status on the overall message. The values that Status may assume are:

STATUS:
SUCCESS = 1
VERSION = 2 Used by the FAS to indicate it does not support version of LFAP used by the CCE.

Reserved is reserved for future expansion and Must Be Zero in present version

Message ID is used to associate each original message with its corresponding response and must be unique for the combination of sender and responder while an original message is pending.

The Message Length excludes the 8 octets of the message header.

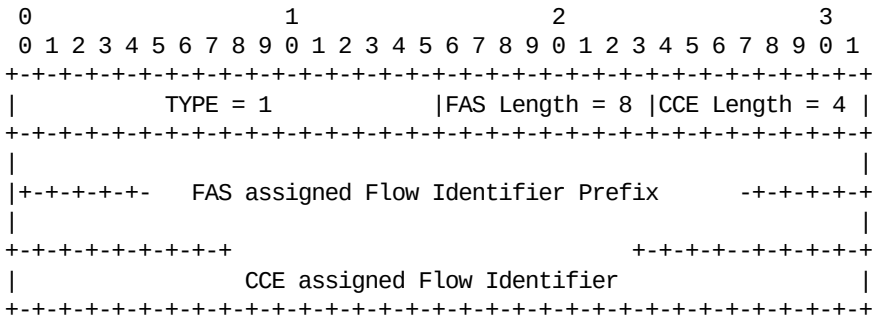
4.0 Information Elements (IE's) Formats

IE fields consist of 2-octet TYPE, 2-octet LENGTH and a variable length VALUE sub-fields. All IEs are even multiples of 4 octets in length, left-aligned and zero filled if necessary. Length is computed excluding the 4 octet TYPE and LENGTH fields.

Individual IEs are formatted as described in following sections.

4.1 Flow ID IE

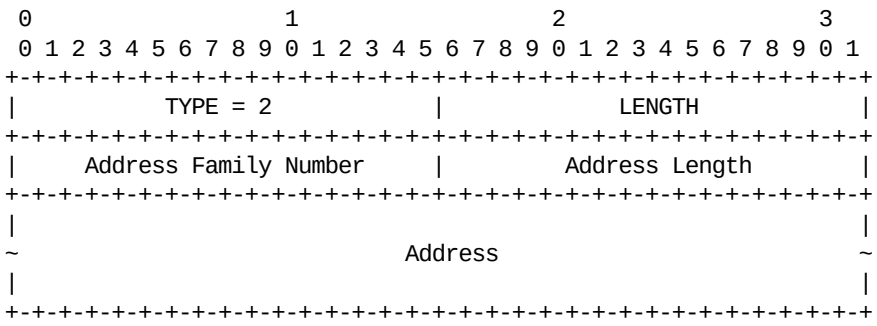
In order to accumulate the flow accounting statistics across multiple FAS's in case of a FAS failure a globally unique flow identifier needs to be formed. To accomplish this the FAS assigns a prefix if requested by the CCE. The CCE then assigns a CCE flow identifier that it guaranties to be unique for the use of the FAS flow identifier prefix for each flow admitted. If the CCE needs to reuse a CCE flow ID it must acquire a new FAS prefix. If a CCE cannot support the FAS flow identifier then it does not request a FAS prefix and uses a FAS length of 0 in all updates to the flow. Flow ID IE is copied exactly in all messages that refer to this flow. IE format is:



The CCE assigned Flow Identifier is a monotonically increasing number. It starts at 1 and increases by 1 until it reaches FFFFFFFF. 0 is not a valid value and is used to represent the lack of a CCE assigned flow identifier.

4.2 Source Address IE

Source address associated with a message. IE format is:



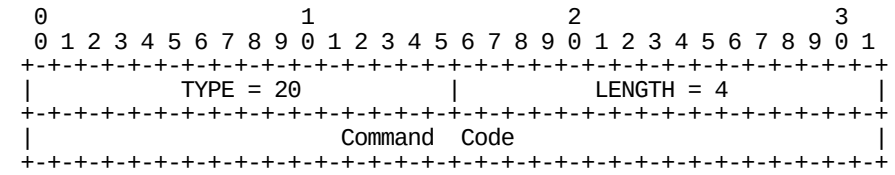
The Address Length field contains the length of the address excluding any pad of zeros used to align the address field.

Address Family Numbers include:

1 - 14	(decimal) as specified in [1700]
15	E.164 with NSAP format subaddress

4.3 Destination Address IE

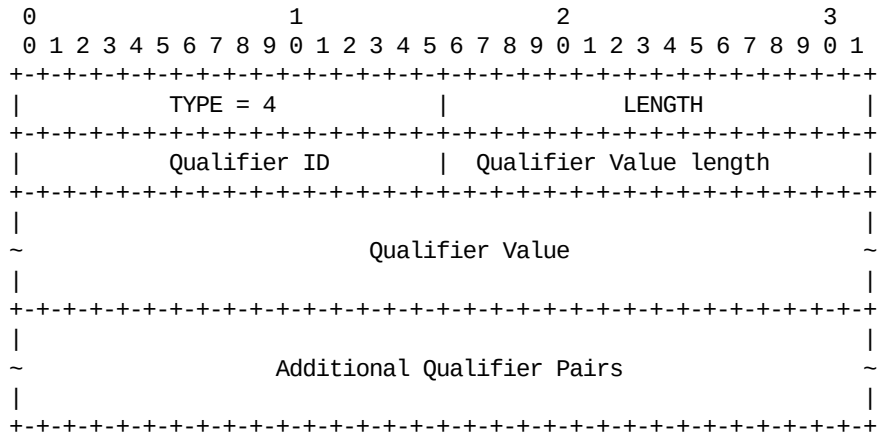
Destination address associated with a message. TYPE is 3, format is as shown for Source Address IE.



4.4 Flow Qualifier IEs

Flow Qualifier IEs contain additional information about a flow. The IE is comprised of a qualifier ID, a length and a value.

Flow Qualifier IE format:



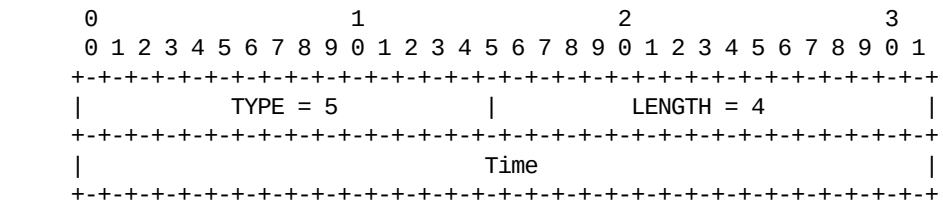
The following qualifier ID's and values are presently defined. Their value length is 4 bytes.

Flow Qualifier	ID	Value	Meaning
Checkpoint	1	1	Checkpoint flow hourly
		2	Checkpoint flow Daily
		3	Checkpoint flow weekly
		4	Checkpoint flow monthly
		5	Checkpoint only at the flow's end
Connection priority	2	100	Priority of this connection is Premium
		200	Priority of this connection is High
		300	Priority of this connection is Medium
		400	Priority of this connection is Low

A switch may map several of its queues into a Premium, High, Medium or Low category

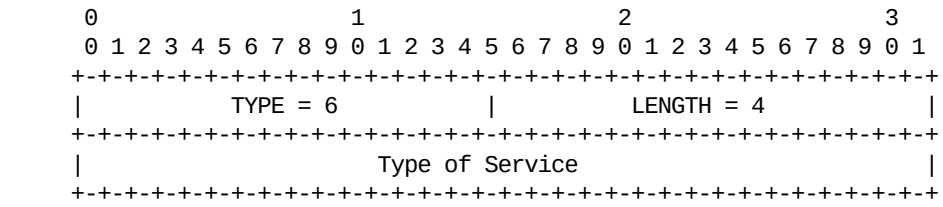
4.5 Time IE

The time (as a SNMPv2 TimeStamp [1443]) associated with the status/statistics observed. TYPE is 5 and format is:



4.6 Type of Service IE

The type of service associated with a message (as defined by rfc 791). TYPE is 6 and format is:

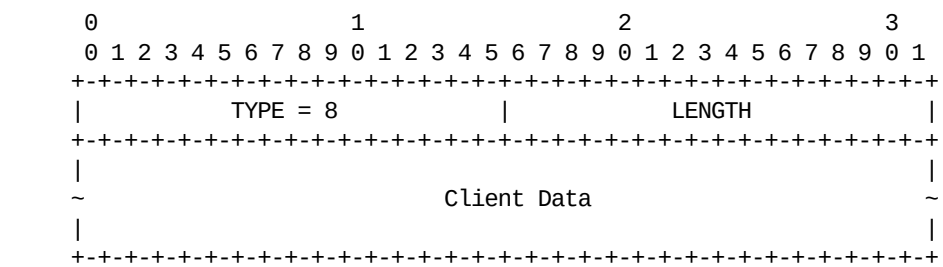


4.7 Source CCE Address IE

Source CCE address is the address of the CCE reporting the flow, TYPE is 7. Format is as shown for FAS IP Address IE. This information is used by applications to later correlate the ingress/egress port with a specific CCE.

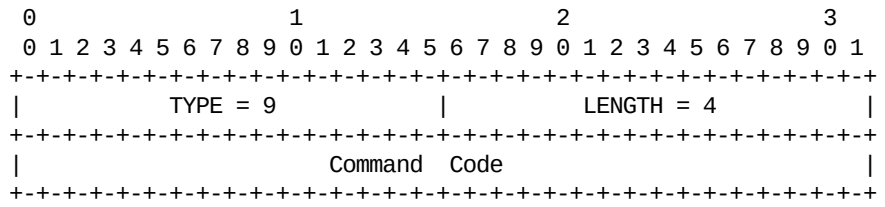
4.8 Client Data IE

For use in determination of accounting policy relative to a specific connection request based on arbitrary client data (OCTET STRING [8824]). IE format is:



4.9 Command Code IE

Command Code is a administrative request command to perform a particular function. IE format is:

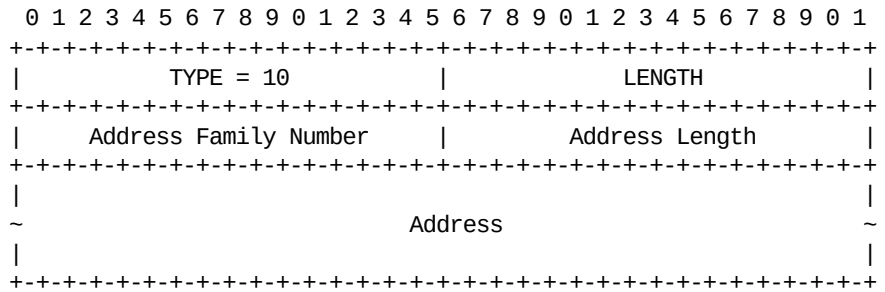


Command code has one of the following values and meanings:

1 - RETURN_INDICATED_FLOWS	-	Used by the FAS to give CCE a list of Flow IE's that it wants the CCE to return FUNs on.
2 - RETURN_FLOW_PREFIX	-	Used by the CCE to request a new flow prefix.
3 - CONNECTION_ACCEPTED	-	Used by the FAS to indicate to the CCE that it will accept it's flow information.
4 - CONNECTION_REJECTED	-	Used by the FAS to indicate to the CCE that it will not accept it's flow information. If the connection is rejected the CCE must go to the next FAS in its list. This will insure that the CCE will not loop infinitely on a subset of the FAS's in its list.
5 - LIST_OF_FASS	-	Used by the CCE to tell the FAS the list of FAS's that the CCE has been given. This list must be given in the order that the CCE would search.
6 - DISCONNECT	-	Used by the FAS to tell the CCE to disconnect from this FAS and connect to the FAS given in this AR message or if none given the CCE may pick any FAS in its list.
7 - KEEPALIVE	-	Sent by the FAS so the CCE can determine if the connection is still intact. These messages are sent at a regular interval. The CCE should treat any message received from the FAS as having the same semantics as receiving the KEEPALIVE AR message. NOTE: the ARA is optional.

4.10 FAS IP Address IE

This is the IP Address of a FAS. The CCE uses this IE in the List of FAS's AR message. IE format is:

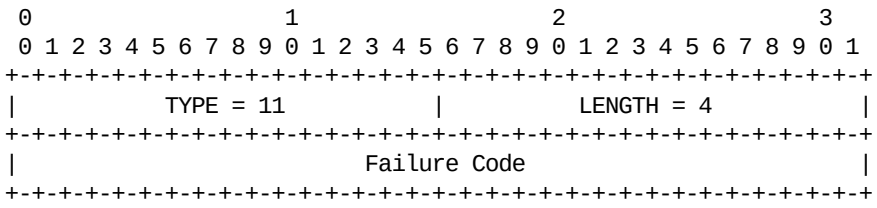


The Address Length field contains the length of the address excluding any pad of zeros used to align the address field.

Address Family Numbers include:
1 = IP (IP Version 4) as specified in RFC 1700
2 = IP6 (IP Version 6) as specified in RFC 1700

4.11 Failure Code IE

Failure code is the reason why an operation failed. IE format is:

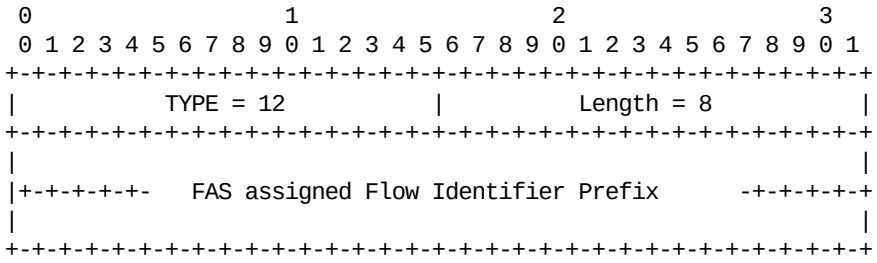


Failure code has one of the following values and meanings:

1 - NO_SUCH_FLOW - The specified flow was not found

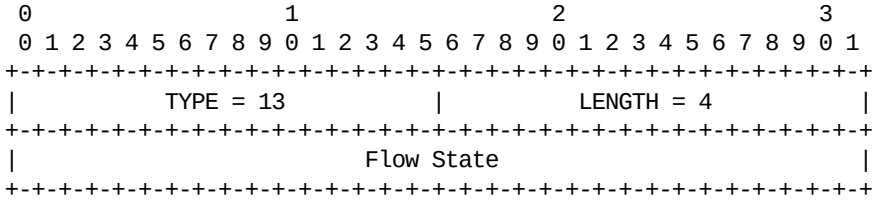
4.12 Flow Identifier Prefix IE

The flow Identifier prefix IE gives the prefix that the FAS has created to maintain a globally unique ID. IE format is:



4.13 Flow State IE

Flow state is the intended end state for the Flow associated with the message containing this IE. IE format is:

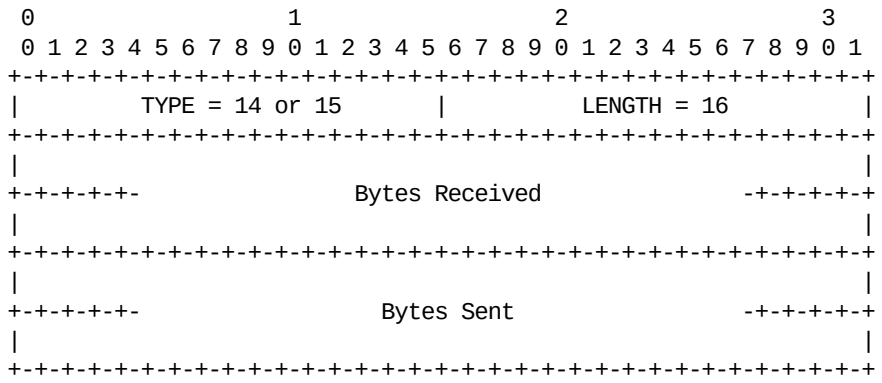


Flow state has one of the following values and meanings:

1 - INACTIVE - Flow is inactive
2 - ACTIVE - Flow is active

4.14 Byte Count IE

Contains the count of octets sent and received associated with the identified connection. IE format is:

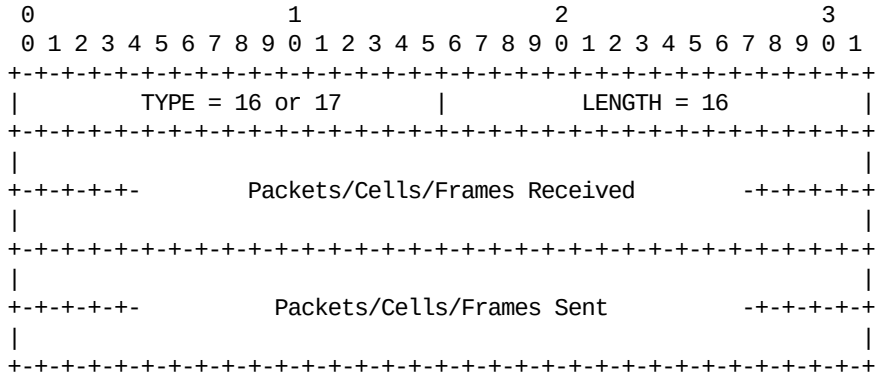


Type 14 Means that the byte count is a running counter and is the count from the beginning of the flow establishment.

Type 15 Means that the byte count is a delta counter and is the count since the last FUN message.

4.15 Packet Count IE

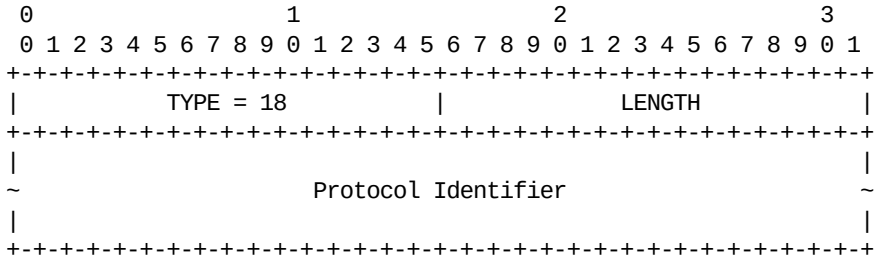
Contains the count of packets cells or frames sent and received associated with the identified connection. IE format is:



Type 16 Means that the packet/cell/frame count is a running counter and is the count from the beginning of the flow establishment.
Type 17 Means that the packet/cell/frame count is a delta counter and is the count since the last FUN message.

4.16 Protocol Identifier IE

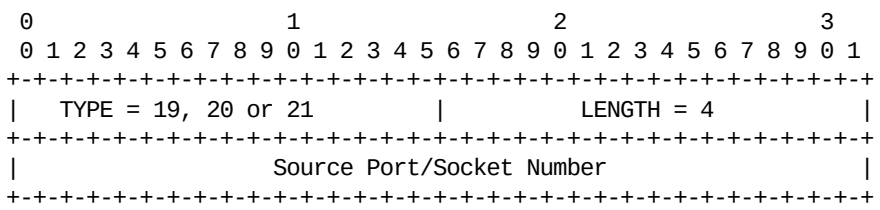
Used in determination of accounting policy relative to a specific connection request based on service type. Protocol Identifier is specified as an OCTET STRING [8824]. The Protocol identifier is composed of the protocolDirID and protocolDirParameters as defined in RFC 2021. The protocol identifier encoding is enumerated in RFC 2074. The Protocol identifier uses the protocolDirTable INDEX Format defined in RFC 2074.



4.17 Source Port IE

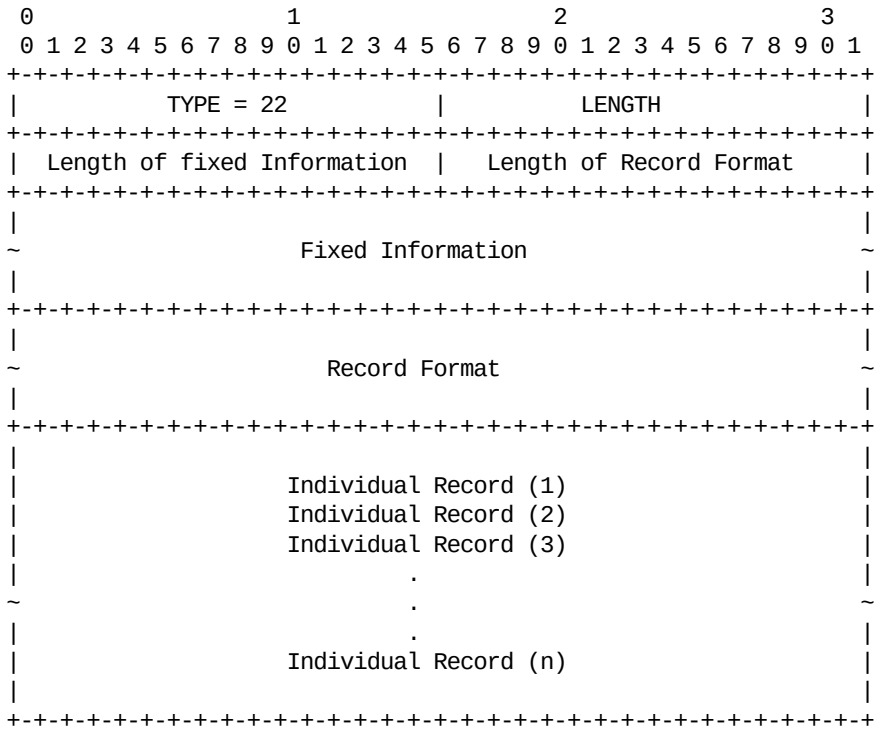
In the case of TCP, UDP and IPX, the protocol IE may not be enough to specify the next layer protocol. rfc 2074 only uses the destination port/socket number. However, it may be the source port/socket number which identifies the protocol. This IE is used to report source port/socket number can be specified (note the destination port/socket is part of the protocol IE). TYPE 19 is UDP source port [see rfc 768], TYPE 20 is TCP source port [see rfc 793] and TYPE 21 is IPX source socket [The IPX protocol is defined by the Novell Corporation. A complete description of IPX may be secured at the following address:

> Novell, Inc.
> 122 East 1700 South
> P. O. Box 5900
> Provo, Utah 84601 USA
> 800 526 5463
> Novell Part # 883-000780-001]



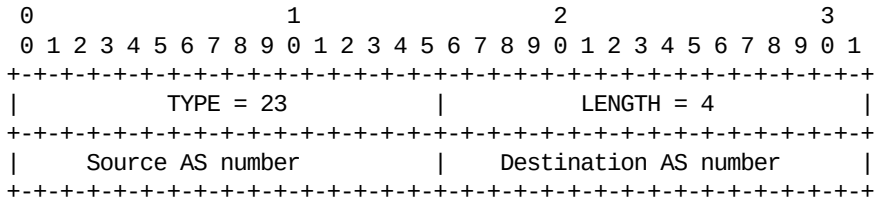
4.18 Multiple Record IE

The Multiple Record IE is composed of 4 parts. The record descriptor, fixed information, record format IEs and individual records. The record descriptor consist of two fields the first field is the length of the fixed information field. The second is the length of the Record format section. The fixed information is the IE's that apply to all the records that follow. The Record Format is the list of IE's that make up each record. The individual record section contains the individual records that are being reported in the format given by the Record Format section.



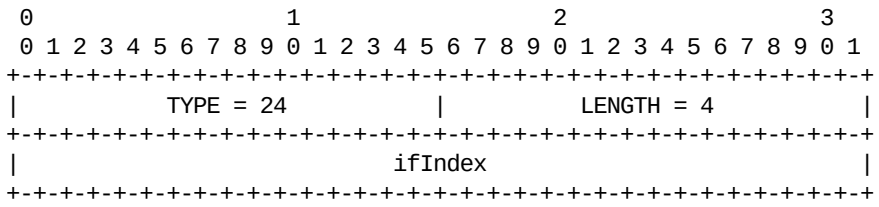
4.19 Source and Destination AS IE

The Autonomous System (AS) numbers for the source IP and the destination IP are provided in this IE. Autonomous System (AS) number is defined by RFC 1930 and RFC 1771 (BGP-4):



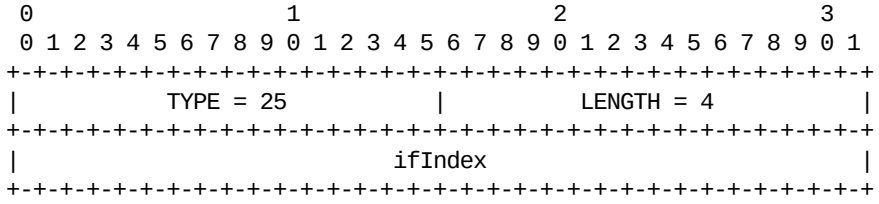
4.20 Ingress Port IE

The ingress ifIndex for the flow is provided in this IE. ifIndex is defined by RFC 2233:



4.21 Egress Port IE

The egress ifIndex for the flow is provided in this IE. ifIndex is defined by RFC 2233:



4.22 4Numeric List of IE's

The numeric list of IE's is as follows

1 Flow ID IE
2 Source Address IE
3 Destination Address IE
4 Flow Qualifier IE's
5 Time IE
6 Type Of Service IE
7 Source CCE Address IE
8 Client Data IE
9 Command Code IE
10 FAS IP Address IE
11 Failure Code IE
12 Flow Identifier Prefix IE
13 Flow State IE
14 + 15 Byte Count IE
16 + 17 Packet Count IE
18 Protocol Identifier IE
19+20+21 Source Port IE
22 Multiple Record IE
23 Source and Destination AS IE
24 Ingress port IE
25 Egress port IE

5.0 Individual Message contents

5.1 Flow Accounting Request (FAR) Message

Status is set to SUCCESS in FAR messages. In addition, FAR messages may contain the following IEs:

Flow ID IE	-	locally unique identifier for Flow (unique to update reporting entity)
Source Address IE	-	flow "from" address
Destination Address IE	-	flow "to" address
Time IE	-	switch time of accounting request
Source CCE Address IE	-	address of the CCE recording the flow
Protocol Identifier IE	-	identifier for protocol used
Source Port IE	-	flow "from" port
Type of Service IE	-	IP ToS
Flow Qualifier IE	-	Connection priority and checkpoint interval
Source and Destination AS IE	-	The AS numbers for the Source IP and Destination IP
Ingress Port IE	-	The ifindex of the Ingress port
Egress Port IE	-	The ifindex of the egress port
Client Data IE	-	client data as applied to this request

Mandatory IE's
- Flow ID
- Source Address
- Destination Address
- Time

Optional IE's
- Protocol Identifier IE
- Source port IE
- Type Of Service IE
- Source CCE Address
- Flow Qualifier IE
- Source and Destination AS IE
- Ingress Port IE
- Egress Port IE
- Client Data

FAR messages are sent by a switch's CCE when it seeks to record a flow that is about to be established. FAR messages refer to a single flow only and do not use multi IE functionality. Protocol ID is the protocol, to the highest layer decoded by the CCE, used by flow. The client data is arbitrary information about the client associated with the flow.

5.2 Flow Update Notification (FUN) Message

Status is set to SUCCESS in FUN messages. In addition, FUN messages may contain the following IEs:

Time IE	-	switch time of notification
Flow ID IE	-	identifier for the flow
Flow State IE	-	state of the flow at time of notification
Byte Count IE	-	octets sent and received for this flow
Packet Count IE	-	packets sent and received for this flow

Mandatory IE's

Flow ID		
Time	-	If multiple IE, only needs to be given once in fixed information section. If given in record format must be in each individual record.

Optional IE's at least one must be present

Flow State
Byte Count
Packet Count

Time IE - switch time of notification
Flow ID IE - identifier for the flow
Flow State IE - state of the flow at time of notification
Byte Count IE - octets sent and received for this flow
Packet Count IE - packets sent and received for this flow

FUN messages are sent periodically (as specified in the switch configuration) by a CCE to the FAS. If it is only a single flow being reported on then just the IE's and their values are present. If multiple flows are to be reported on then the multiple record IE should be used. This results in reduced overhead on transmissions. FUN messages may also be sent as a result of an AR.

The flow ID identifies the flow to which this update applies. Flow state is the state of the flow at the time this message is sent. Counts are as specified in the IE definitions. The FAS's are coordinated and will resolve the reception of FUN information from CCE who has lost connection with its FAS and has gone to an alternative FAS.

5.3 Administrative Request (AR) Message

AR messages may contain the Command IEs:

Mandatory IE's
Command IE

Optional IE
Flow ID
FAS IP Address

AR messages are sent by either the switch CCE or the FAS when they seek to perform one of the Command IE's.

5.4 Administrative Request Acknowledge (ARA) Message

Status reflects the result of the corresponding AR message (see Error Handling for details). Message ID is copied from the AR message. In addition, ARA messages may have the following IEs:

Flow ID IE - if FAS requested statistics on an unknown flow.
Failure Code - for the Flow ID IE above.

Flow Identifier Prefix IE - if the ARA is the response to a CCE Command to RETURN_FLOW_PREFIX.

ARA messages are sent by a FAS or CCE in response to AR message received by a CCE or FAS respectively. The ARA is optional for an AR KEEPALIVE message.

5.5 Version Request (VR) Message

VR messages do not contain any IEs.

VR messages are sent by the CCE to request a session with the FAS under a specific version of the LFAP protocol.

5.6 Version Request Acknowledge (VRA) Message

VRA messages do not contain any IEs.

Message ID is copied from the VR message. If the FAS supports the version requested in the VR message it also copies the version number from the VR message to the VRA message. At this point protocol version negotiation is complete and a conversation can begin.

If the FAS does not support the version requested in the VR message, the FAS places the highest version it supports in the message header and sets the STATUS field to VERSION, which indicates a version error.

6.0 Session Establishment

Session establishment happens in two phases. In the first phase the protocol version is established. In the second phase, the FAS allows or denies the connection. The FAS may deny the connection for a variety of reasons. For example, the CCE may not be authorized to connect to this FAS.

6.1 Protocol Negotiation

In this phase the FAS and the CCE negotiate the version of the LFAP protocol to be used. The CCE initiates the process by sending a VR message with the highest version it supports in the message header. If the FAS supports the version it will respond with a VRA with the same version number and a status of success. For example, a CCE may request version 3 and the FAS may support version 3, 4, and 5. In this case the FAS will respond with a VRA message with a version of 3 and status of success.

If the FAS does not support the version it will send a VRA message with the highest version it supports and the status set to VERSION. At this point the CCE may either disconnect and try another FAS. Or, if the CCE supports the version indicated by the FAS, the CCE may send a new VR message with the indicated version number. For example, if the CCE supports version 5 and 6, it will send a VR message to the FAS with version 6. Since the FAS supports version 3, 4, and 5, it responds with a VRA message with a version of 5 and a status of VERSION (indicating it does not support version 6). The CCE may now send another VR message, this time with version 5.

6.2 Connection Establishment

Once the protocol negotiation is complete, the FAS then must either accept or reject the connection. First, the FAS must receive the list of FASS AR message from the CCE. At this point the FAS will either accept/reject the connection with an AR message or send it to another FAS by sending the DISCONNECT message.

NOTES:

1. The normal sequence of events when a CCE and FAS start to talk would be as follows:
   - The CCE opens a TCP socket.
   - The CCE sends a VR message with the desired protocol version
   - The FAS sends a VRA response with its desired version.
   - Assuming the two agree, the CCE then sends an AR LIST_OF_FASS and the FAS responds with and ARA
   - The FAS then sends an AR CONNECT_ACCEPTED
   - The CCE sends an ARA response

   From this point on order is not guaranteed.
   - The CCE sends AR RETURN_FLOW_PREFIX if needed.
   - The CCE sends FAR's/FUN's (etc.).

7.0 Error Handling

Corrupted message contents - Receipt of a LFAP message which cannot be understood, will be ignored by the receiver. If possible, an error should be logged is possible. Other error handling is naturally associated with each message and is listed in the following sections.

7.1 VRA Related Error Handling

If the FAS does not support the version requested in the VR message, the FAS places the highest version it supports in the message header and sets the STATUS field to VERSION, which indicates a version error.

7.2 ARA Related Error Handling

Flow statistics requested for a non-existent flow - The Flow ID IE identified a connection for which this CCE has no state information. The ARA message has the Flow ID and a Failure Code set to NO_SUCH_FLOW and contains the Flow ID copied from the corresponding AR message. If there were multiple flows that were non-existent then the multi-ie format could have the Failure Code in the fixed information section and the individual Flow ID's in the record section.

8.0 Security Considerations

Security is not addressed in this document.

9.0 Author's Addresses

Paul Calato
Phone: +1 (603) 337-7625
Email: calato@ctron.com

Dana Cook
Phone: +1 (603) 337-7551
Email: dcook@cabletron.com

Todd Crowley
Phone: +1 (603) 337-7173
Email: crowley@cabletron.com

Cabletron Systems, Inc. is located at:
P.O. Box 5005
Rochester, NH, 03866-5005
USA

10.0 References

[363] "B-ISDN ATM Adaptation Layer (AAL) Specification," International Telecommunication Union, ITU-T Recommendation I.363, Mar. 1993.

[1443] "Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1443, April 1993.

[1700] "Assigned Numbers," RFC 1700, October 1994.

[8824] Information technology - Open Systems Interconnection - "Specification of Abstract Syntax Notation One (ASN.1),

Second edition", ISO/IEC TR 8824: 1990 (E) 1990-12-15.

[9577] "Telecommunications and Information Exchange Between Systems - Protocol Identification in the Network Layer", ISO/IEC TR 9577: 1990 (E) 1990-10-15.

[LANE] "LAN Emulation Over ATM Specification - Version 1.0", ATM Forum af-lane-021.000, January, 1995.

[2021] "Remote Network Monitoring Management Information Base Version 2" RFC 2021, January 1977.

[2074] "Remote Network Monitoring MIB Protocol Identifiers" RFC 2074, January 1997.