SIPCLF G. Salgueiro
Internet-Draft Cisco Systems
Intended status: Standards Track V. Gurbani
Expires: June 24, 2013 Bell Labs, Alcatel-Lucent
A. B. Roach
Tekelec
December 21, 2012

Format for the Session Initiation Protocol (SIP) Common Log Format (CLF)
draft-ietf-sipclf-format-11

Abstract

The SIPCLF Workgroup has defined a common log format framework for Session Initiation Protocol (SIP) servers. This common log format mimics the successful event logging format found in well-known web servers like Apache and web proxies like Squid. This document proposes an indexed text encoding format for the SIP Common Log Format (CLF) that retains the key advantages of a text-based format, while significantly increasing processing performance over a purely text-based implementation. This file format adheres to the SIP CLF information model and provides an effective encoding scheme for all mandatory and optional fields that appear in a SIP CLF record.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 3
3. Document Conventions 4
4. Format 5
4.1. Index Pointers 8
4.2. Mandatory Fields 10
4.3. SIP CLF Encoding and Character Escaping Requirements . . . 14
4.4. Optional Fields 15
5. Example SIP CLF Record 24
6. Text Tool Considerations 26
7. Security Considerations 26
8. Operational Guidance 26
9. IANA Considerations 26
9.1. SIP CLF Version 27
9.2. SIP CLF Transport Flag 27
10. Acknowledgements 28
11. References 28
11.1. Normative References 28
11.2. Informative References 29
Authors' Addresses 30

1. Introduction

The extensive list of benefits and the widespread adoption of the Apache Common Log Format (CLF) has prompted the development of an analogous event logging mechanism for the Session Initiation Protocol [RFC3261] (SIP). Implementing a logging scheme for SIP is a considerable challenge. This is due in part to the fact that the behavior of a SIP entity is more complex as compared to an HTTP entity. Additionally, there are shortcomings to the purely text-based HTTP Common Log Format that need to be addressed in order to allow for real-time inspection of SIP log files [I-D.ietf-sipclf-problem-statement]. Experience with Apache Common Log Format has shown that dealing with large quantities of log data can be very processor intensive, as doing so necessarily requires reading and parsing every byte in the log file(s) of interest.

An implementation independent framework for the SIP CLF has been defined in [I-D.ietf-sipclf-problem-statement]. This memo describes an indexed text file format for logging SIP messages received and sent by SIP clients, servers, and proxies that adheres to the information model presented in Section 8 of [I-D.ietf-sipclf-problem-statement]. This document defines a format that is no more difficult to generate by logging entities than standard (i.e., non-indexed) text log formats, while being radically faster to process. In particular, the format is optimized for both rapidly scanning through log records, as well as quickly locating commonly accessed data fields.

Further, the format proposed by this document retains the key advantage of being human readable and able to be processed using the various Unix text processing tools, such as sed, awk, perl, cut, and grep.

2. Terminology

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].

"SHOULD", "SHOULD NOT", "RECOMMENDED", and "NOT RECOMMENDED" are appropriate when valid exceptions to a general requirement are known to exist or appear to exist, and it is infeasible or impractical to enumerate all of them. However, they should not be interpreted as permitting implementors to fail to implement the general requirement when such failure would result in interoperability failure.

[RFC3261] defines additional terms used in this document that are

specific to the SIP domain such as "proxy"; "registrar"; "redirect server"; "user agent server" or "UAS"; "user agent client" or "UAC"; "back-to-back user agent" or "B2BUA"; "dialog"; "transaction"; "server transaction".

This document uses the term "SIP Server" that is defined to include the following SIP entities: user agent server, registrar, redirect server, a SIP proxy in the role of user agent server, and a B2BUA in the role of a user agent server.

The reader is expected to be familiar with the terminology and concepts defined in [I-D.ietf-sipclf-problem-statement].

3. Document Conventions

This document defines the logging syntax for the SIP CLF. This syntax is demonstrated through the use of various examples. The formatting described here does not permit these examples to be unambiguously rendered due to the constraints imposed by the formatting rules for RFCs. To avoid ambiguity and to meet the RFC layout requirements this document uses the <allOneLine/> markup convention established in [RFC4475].

For the sake of clarity and completeness, the entire text defining this markup convention from Section 2.1 of [RFC4475] is quoted below:

> Several of these examples contain unfolded lines longer than 72 characters. These are captured between <allOneLine/> tags. The single unfolded line is reconstructed by directly concatenating all lines appearing between the tags (discarding any line feeds or carriage returns). There will be no whitespace at the end of lines. Any whitespace appearing at a fold-point will appear at the beginning of a line.
>
> The following represent the same string of bits:

```
      Header-name: first value, reallylongsecondvalue, third value

      <allOneLine>
      Header-name: first value,
       reallylongsecondvalue
      , third value
      </allOneLine>

      <allOneLine>
```

```
         Header-name: first value,
          reallylong
         second
         value,
          third value
         </allOneLine>
```

Note that this is NOT SIP header-line folding, where different strings of bits have equivalent meaning.

The IP addresses used in the examples in this document correspond to the documentation address block 192.0.2.0/24 (TEST-NET-1) as described in [RFC5737].

4. Format

The Common Log Format for the Session Initiation Protocol [I-D.ietf-sipclf-problem-statement] defines an information model to which this logging format format adheres, and Section 8.1 of that document defines all the mandatory information model elements.

This document defines the format of SIP CLF records as follows:

```
     0          7 8        15 16       23 24         31
     +-----------+-----------+-----------+-----------+
     |  Version  |           Record Length           | 0 - 3
     +-----------+-----------+-----------+-----------+
     |       Record Length (cont)        |    0x2C   | 4 - 7
     +-----------+-----------+-----------+-----------+
     |              CSeq Pointer (Hex)               | 8 - 11
     +-----------+-----------+-----------+-----------+
     |      Response Status-Code Pointer (Hex)       | 12 - 15
     +-----------+-----------+-----------+-----------+
     |              R-URI Pointer (Hex)              | 16 - 19
     +-----------+-----------+-----------+-----------+
     |   Destination IP address:port Pointer (Hex)   | 20 - 23
     +-----------+-----------+-----------+-----------+
     |     Source IP address:port Pointer (Hex)      | 24 - 27
     +-----------+-----------+-----------+-----------+
     |             To URI Pointer (Hex)              | 28 - 31
     +-----------+-----------+-----------+-----------+
     |             To Tag Pointer (Hex)              | 32 - 35
     +-----------+-----------+-----------+-----------+
     |            From URI Pointer (Hex)             | 36 - 39
     +-----------+-----------+-----------+-----------+
     |            From Tag Pointer (Hex)             | 40 - 43
```

```
     +-----------+-----------+-----------+-----------+
     |             Call-Id Pointer (Hex)             | 44 - 47
     +-----------+-----------+-----------+-----------+
     |           Server-Txn Pointer (Hex)            | 48 - 51
     +-----------+-----------+-----------+-----------+
     |           Client-Txn Pointer (Hex)            | 52 - 55
     +-----------+-----------+-----------+-----------+
     |      Optional Fields Start Pointer (Hex)      | 56 - 59
     +-----------+-----------+-----------+-----------+
     |    0x0A   |                                   | 60 - 63
     +-----------+                                   +
     |                   Timestamp                   | 64 - 67
     +                                   +-----------+
     |                                   |    0x2E   | 68 - 71
     +-----------+-----------+-----------+-----------+
     |         Fractional Seconds        |    0x09   | 72 - 75
     +-----------+-----------+-----------+-----------+
     |                  Flags Field                  | 76 - 79
     +-----------+-----------+-----------+-----------+
     |Flag (cont)|    0x09   |                       | 80 - 83
     |-----------+-----------+                       |
     |                                               |
     |                                               |
     |      Mandatory Fields (variable length)       |
     |                                               |
     |                                               |
     +-----------+-----------+-----------+-----------+
     |    0x09   |          Tag          |    0x40   |\
     +-----------+-----------+-----------+-----------+ \
     |                   Vendor-ID                   |  \
     +-----------+-----------+-----------+-----------+   \
     |                Vendor-ID (cont)               |    \   Repeated
     +-----------+-----------+-----------+-----------+     \  as many
     |     0x2C  |            Length (Hex)           |      > times as
     +-----------+-----------+-----------+-----------+     /  necessary
     | Len (cont)|    0x2C   |    BEB    |    0x2C   |    /
     +-----------+-----------+-----------------------|   /
     |                                               |  /
     |            Value (variable length)            | /
     |                                               |/
     +-----------+-----------+-----------+-----------+
     |    0x0A   |
     +-----------+
```

Figure 1: SIP Common Log Format

The format presented in Figure 1 is for a single SIP CLF log entry. While there is no actual subdivision in practice, this format can be

logically subdivided into the following three distinct components:

1. Index Pointers - The first 60-bytes of this format. This portion is metadata, primarily composed of a list of pointers that indicate the beginning of both the variable length mandatory and optional fields that are logged as part of this record. These pointers are implemented as a mechanism to improve processing of these records and to allow a reader to expeditiously skip directly to the desired field without unnecessarily going through the entire record. This logical subdivision within the SIP CLF format will be referenced in this document with the <IndexPointers> tag. A 0x0A (LF character) delimits <IndexPointers> from the next logical grouping.

2. Mandatory Fields - The next logical grouping in this format is a tab (0x09) delimited listing of the mandatory fields as described in Section 8.1 of [I-D.ietf-sipclf-problem-statement] and in the order listed in <IndexPointers>. This logical subdivision within the SIP CLF format will be referenced in this document with the <MandatoryFields> tag.

3. Optional Fields - The last logical component MAY be present as it is an OPTIONAL extension to the SIP CLF format. Its purpose is to provide flexibility to the developer of this SIP CLF to log any desired fields not included in <MandatoryFields>. This includes SIP bodies and any vendor-specific extensions. This logical subdivision within the SIP CLF format will be referenced in this document with the <OptionalFields> tag.

This logical structure of the SIP CLF record format can be graphically represented as shown in Figure 2 below:

```
                              <IndexPointers>
                              <MandatoryFields>
                              <OptionalFields>
```

Figure 2: Logical Structure of the SIP CLF Record

Note that Figure 1 and Figure 2 plus the terminating line-feed (0x0A) at the end of the SIP CLF record are different representations of the same format but are functionally equivalent. The representation of this format is a two line record where the <IndexPointers> metadata is on one line and the actual data like <MandatoryFields> and <OptionalFields> (if present) is on another.

In the following sections note that indications of "hexadecimal

encoded" indicate values that are always unsigned and are to be written out in human-readable base-16 numbers using the UTF-8 characters 0x30 through 0x39 ('0' through '9') and 0x41 through 0x46 ('A' through 'F'). Similarly, indications of "decimal encoded" indicate that the value is to be written out in human readable base-10 number using the UTF-8 characters 0x30 through 0x39 ('0' through '9'). In both encodings, numbers always take up the number of bytes indicated, and are padded on the left with UTF-8 '0' (zero) characters to fill the entire space.

4.1. Index Pointers

The <IndexPointers> portion of the SIP CLF record (shown in Figure 3) is a 60-byte header that indicates metadata about the record.

```
         0          7 8        15 16       23 24         31
         +-----------+-----------+-----------+-----------+
         |  Version  |           Record Length           | 0 - 3
         +-----------+-----------+-----------+-----------+
         |       Record Length (cont)        |    0x2C   | 4 - 7
         +-----------+-----------+-----------+-----------+
         |              CSeq Pointer (Hex)               | 8 - 11
         +-----------+-----------+-----------+-----------+
         |      Response Status-Code Pointer (Hex)       | 12 - 15
         +-----------+-----------+-----------+-----------+
         |              R-URI Pointer (Hex)              | 16 - 19
         +-----------+-----------+-----------+-----------+
         |   Destination IP address:port Pointer (Hex)   | 20 - 23
         +-----------+-----------+-----------+-----------+
         |     Source IP address:port Pointer (Hex)      | 24 - 27
         +-----------+-----------+-----------+-----------+
         |             To URI Pointer (Hex)              | 28 - 31
         +-----------+-----------+-----------+-----------+
         |             To Tag Pointer (Hex)              | 32 - 35
         +-----------+-----------+-----------+-----------+
         |            From URI Pointer (Hex)             | 36 - 39
         +-----------+-----------+-----------+-----------+
         |            From Tag Pointer (Hex)             | 40 - 43
         +-----------+-----------+-----------+-----------+
         |             Call-Id Pointer (Hex)             | 44 - 47
         +-----------+-----------+-----------+-----------+
         |           Server-Txn Pointer (Hex)            | 48 - 51
         +-----------+-----------+-----------+-----------+
         |           Client-Txn Pointer (Hex)            | 52 - 55
         +-----------+-----------+-----------+-----------+
         |      Optional Fields Start Pointer (Hex)      | 56 - 59
         +-----------+-----------+-----------+-----------+
```

Figure 3: Index Pointers

The fields that make up <IndexPointers> are described below:

Version (1 byte): UTF-8 encoded version for the SIP CLF record. Range of valid values for the Version is from 'A' (0x41) to 'Z' (0x5A). This document uses a Version value of "0x41" ('A').

The value of the SIP CLF Version MUST be incremented for any new SIP CLF specification that changes any part of the SIP CLF record format. The SIP CLF Version values are IANA-assigned (Section 9.1) via the Standards Action method as described in [RFC5226].

Since the version is specified per record it is possible that a SIP CLF log file could contain records with different versions. Under normal operating conditions this is an unlikely occurrence and SHOULD be avoided if possible.

Record Length (6 bytes): Hexadecimal encoded total length of this log record, beginning with the "Version" octet and ending with the terminating line-feed.

Bytes 8 through 55 contain hexadecimal encoded pointers that point to the starting location of each of the variable-length mandatory fields. Bytes 56 through 59 contain hexadecimal encoded pointer that points to the starting location of the optional fields portion of the SIP CLF record. Note that there are no delimiters between these pointer values -- they are packed together as a single, 52-character hexadecimal encoded string. The "Pointer" fields indicate absolute byte values within the record, and are therefore >=82. They point to the start of the corresponding value within the <MandatoryFields> portion. A description of each of the mandatory fields that these pointer values point to can be found in Section 4.2.

Optional Fields Start Pointer: This final pointer indicates the location within the SIP CLF record where the OPTIONAL group of <OptionalFields> begin, if present. The "Optional Fields Start Pointer" points to the UTF-8 Tab (0x09) character for the first entry in the <OptionalFields> portion. If the OPTIONAL group of <OptionalFields> are not implemented, then the "Optional Fields Start Pointer" field MUST point to the terminating line-feed (0x0A) at the end of the SIP CLF record.

4.2. Mandatory Fields

The <MandatoryFields> portion of the SIP CLF record is shown below:

```
         0          7 8        15 16       23 24         31
         +-----------+-----------+-----------+-----------+
         |    0x0A   |                                   | 60 - 63
         +-----------+                                   +
         |                   Timestamp                   | 64 - 67
         +                                   +-----------+
         |                                   |    0x2E   | 68 - 71
         +-----------+-----------+-----------+-----------+
         |         Fractional Seconds        |    0x09   | 72 - 75
         +-----------+-----------+-----------+-----------+
         |                  Flags Field                  | 76 - 79
         +-----------+-----------+-----------+-----------+
         |Flag (cont)|    0x09   |                       | 80 - 83
         |-----------+-----------+                       |
         |                                               |
         |                                               |
         |      Mandatory Fields (variable length)       |
         |                                               |
         |                                               |
         +-----------+-----------+-----------+-----------+
```

Figure 4: Mandatory Fields

Following the pointers in <IndexPointers>, two fixed-length fields are encoded to specify the exact time of the log entry. As before, all fields are completely filled, pre-pending values with '0' characters as necessary.

Timestamp (10 bytes): Decimal encoded date and time of the request or response represented as the number of seconds since the Unix epoch (i.e. seconds since midnight, January 1st, 1970, GMT).

Fractional Seconds (3 bytes): Decimal encoded fractional seconds portion of the Timestamp field to millisecond accuracy.

The combined Timestamp and Fractional Seconds fields are represented in the log file as a UTF-8 encoded string representing the date and time of the request or response represented as the number of seconds and milliseconds since the Unix epoch. The number of milliseconds are separated by a "." (UTF-8 character 0x2E) from the number of seconds.

Flags Field (5 bytes):

byte 1 - Request/Response Flag

R = Request
r = Response

byte 2 - Retransmission Flag

O = Original transmission
D = Duplicate transmission
S = Server is stateless [i.e., retransmissions are not detected]

byte 3 - Sent/Received Flag

S = Sent message
R = Received message

byte 4 - Transport Flag

The Transport Flag values are IANA-assigned (Section 9.2) via the IETF Review method as described in [RFC5226]. Currently registered values are:

U = UDP
T = TCP
S = SCTP

byte 5 - Encryption Flag

E = Encrytped message (TLS, DTLS, etc.)
U = Unencrypted message

After the "Timestamp", "Fractional Seconds" and the "Flags" fields are the values for the mandatory fields specified in Section 8.1 of [I-D.ietf-sipclf-problem-statement], which are described below:

CSeq: The Command Sequence header field, including the CSeq number and method name.

Response Status-Code: Set to the value of the SIP response status code for responses. Set to a single UTF-8 dash (0x2D) for requests.

R-URI: The Request-URI in the start line (mandatory in request), including any URI parameters.

Destination IP address:port The IP address of the downstream server and the port number, separated by a single ':'. IPv4 addresses are represented in "dotted decimal" notation as per [RFC1166]. IPv6 addresses are represented using the hexadecimal notation detailed in Section 4 of [RFC5952] (or the special-case mixed hexadecimal and decimal notation detailed in Section 5 of [RFC5952]) and enclosed in square brackets ('[' and ']').

Source IP address:port Source IP address:port The IP address of the upstream client and the port number over which the SIP message was received, separated by a single ':'. IPv4 addresses are represented in "dotted decimal" notation as per [RFC1166]. IPv6 addresses are represented using the hexadecimal notation detailed in Section 4 of [RFC5952] (or the special-case mixed hexadecimal and decimal notation detailed in Section 5 of [RFC5952]) and enclosed in square brackets ('[' and ']').

To URI: Value of the URI in the To header field.

To Tag: Value of the tag parameter (if present) in the To header field.

From URI: Value of the URI in the From header field.

From Tag: Value of the tag parameter (if present) in the From header field.

Call-Id: The value of the Call-ID header field.

Server-Txn: Server transaction identification code - the transaction identifier associated with the server transaction. Implementations can reuse the server transaction identifier (the topmost branch-id of the incoming request, with or without the magic cookie), or they could generate a unique identification string for a server transaction (this identifier needs to be locally unique to the server only.) This identifier is used to correlate ACKs and CANCELs to an INVITE transaction; it is also used to aid in forking. (See Section 9 of [I-D.ietf-sipclf-problem-statement] for usage.)

Client-Txn: Client transaction identification code - this field is used to associate client transactions with a server transaction for forking proxies or B2BUAs. Upon forking, implementations can reuse the value they inserted into the topmost Via header's branch parameter, or they can generate a unique identification string for

the client transaction. (See Section 9 of [I-D.ietf-sipclf-problem-statement] for usage.)

Note: The definitions of the Server-Txn and Client-Txn are taken directly from [I-D.ietf-sipclf-problem-statement] and are provided here only as a convenience to the implementer. The definitions specified in [I-D.ietf-sipclf-problem-statement] should be considered authoritative in the event of a conflict.

This data MUST appear in the order listed in <IndexPointers>, and each field MUST be present. Fields are subject the maximum SIP CLF field size of 4096 bytes as detailed in Section 8 of [I-D.ietf-sipclf-problem-statement].

4.3. SIP CLF Encoding and Character Escaping Requirements

The mandatory fields in a SIP CLF record are separated by a single UTF-8 Tab character (0x09). Any Tab characters present in the data to be written will be replaced by a UTF-8 space character (0x20) prior to being logged.

The decision to replace tabs with spaces was based on there being no standardized use of tabs in SIP headers to convey any other meaning than whitespace. Tabs may appear in message bodies, and in the event that the bodies are logged, the conversion to space may cause problems when reconstructing the body from the corresponding log entry. Two consequences of the decision to replace tab with a space character are: (a) it will become impossible to reconstruct a signature over the logged field that matches the signature over fields in the original SIP message, and (b) any future SIP header fields that include tabs with a different semantic meaning than simply signifying whitespace will lose this meaning when logged. And finally, the tabs to spaces substitution MUST occur when logging mandatory fields and optional SIP Header Field or Reason-Phrase (Tag=00); it MUST also occur when when optionally logging either the entire message (Tag=02) or simply a SIP body (Tag=01) as described in Section 4.4.

An element will not always have an appropriate value to provide for one of these fields, even when the field is required to appear in the SIP CLF record. In such circumstances, when a given mandatory field from Section 4.2 and specified in Section 8.1 of [I-D.ietf-sipclf-problem-statement]) is not present then that empty field MUST be encoded as a single horizontal dash ("-"). In the event that a field failed to parse it MUST be encoded as a single question mark ("?"). If these characters are part of a sequence of other characters, then there is no ambiguity. If the field being logged contains only one character, and that character is the literal

"-", the implementation SHOULD insert an escaped %2D for that field in the SIP CLF record. Similarly, if the field contains only one character, and that character is the literal "?", the implementation SHOULD insert an escaped %3F for that field in the SIP CLF record.

The terminating carriage return line feed (CRLF) after a given header field value MUST NOT be logged. Since a bare CRLF sequence is not permitted within a SIP header field value, mandatory fields MUST NOT contain a CRLF when logged and consequently no escaping mechanism is required for it.

Clearly a SIP parser could not possibly successfully parse a SIP CLF record in its entirety given the SIP CLF format described in this document. It is possible to parse individual fields in the SIP CLF record if they are extracted and given to a SIP parser that would normally parse those sequence of strings. It should be noted that any field value that is modified by the escaping mechanisms defined in this document before logging ('-','?', and CRLF) is likely no longer well-formed SIP and will fail when given to such a parser.

The intent of logging using SIP CLF is not to faithfully recreate the bit-exact SIP message being logged. In fact, the formatting rules, encoding and character escaping requirements preclude this and may introduce information loss relative to the original SIP message. A log reader should never unescape anything in the SIP CLF record since they are intended to be machine processed using text tools such as grep and awk. The human user behind the log reader may be required to infer more semantics about any differences between the original SIP message and its SIP CLF representation.

4.4. Optional Fields

The <OptionalFields> portion of the SIP CLF record is shown below:

```
  0          7 8        15 16       23 24        31
  +-----------+-----------+-----------+-----------+
  |    0x09   |          Tag          |    0x40   |\
  +-----------+-----------+-----------+-----------+ \
  |                   Vendor-ID                   |  \
  +-----------+-----------+-----------+-----------+   \
  |                Vendor-ID (cont)               |    \   Repeated
  +-----------+-----------+-----------+-----------+     \  as many
  |     0x2C  |            Length (Hex)           |      > times as
  +-----------+-----------+-----------+-----------+     /  necessary
  | Len (cont)|    0x2C   |    BEB    |    0x2C   |    /
  +-----------+-----------+-----------------------|   /
  |                                               |  /
  |            Value (variable length)            | /
  |                                               |/
  +-----------+-----------+-----------+-----------+
```

Figure 5: Optional Fields

Optional fields are those SIP message elements that are not a part of the mandatory fields list detailed in Section 8.1 of [I-D.ietf-sipclf-problem-statement]. After the <MandatoryFields> section, there is an OPTIONAL <OptionalFields> group (shown in Figure 5) that MAY appear zero or more times. This <OptionalFields> group provides extensibility to the SIP CLF. It allows SIP CLF implementers the flexibility to extend the logging capability of this indexed text representation beyond just the mandatory log elements described in Section 8.1 of [I-D.ietf-sipclf-problem-statement].

Logging any optional SIP elements MUST be done according to the format shown in Figure 5. The location of the start of <OptionalFields> within the SIP CLF record is indicated by the "Optional Fields Start Pointer" field in <IndexPointers>. After the initial Tab delimiter byte (0x09) shown in Figure 5, the optional field being logged is generally represented by the notation:

Tag@Vendor-ID,Length,BEB,Value

The optional field identifier (Tag@Vendor-ID) is composed of a two byte Tag and an eight byte Vendor-ID (both decimal encoded) separated by an "@" character (0x40). This uniquely identifies the optional field being logged. The format for this identifier is loosely modeled after the private use option used by the Syslog protocol [RFC5424] (Note: this is the second format detailed in Section 6.3.2 of [RFC5424]). It makes use of the Private Enterprise Number (PEN), which provides an identifier through a globally unique name space

[PEN]. This syntax provides the necessary extensibility to SIP CLF to allow logging of any SIP header, body, as well as any vendor-specified SIP element.

The Base64 Encoded Byte (BEB) is a boolean that is used to indicate whether the optional element being logged is Base64 encoded or not. The Value field for the optional element being logged MUST be Base64 encoded if it has any characters that are 'unprintable'. For the purposes of this document, we define 'unprintable' to mean a string of octets that: (a) contains an octet with a value in the range of 0 to 31, inclusive; (b) contains an octet with a value of 127; or (c) contains any series of octets greater than or equal to 128 which do not form a valid UTF-8 sequence, as specified by [UNICODE]. If the optional element being logged is Base64 encoded then BEB=0x01; if it is not Base64 encoded then BEB=0x00.

Optional fields are logged according to the following two syntax rules:

(1) Vendor-ID = 00000000

A Vendor-ID of zero is used to log the entire SIP message, message body, Reason-Phrase or any SIP header fields that are not a part of the mandatory fields list detailed in Section 8.1 of [I-D.ietf-sipclf-problem-statement]. The following Tag values are used to identify which of these optional elements are being logged:

Tag = 00 - Log SIP Header Field or Reason-Phrase

When logging a SIP Header Field (Tag=00), the associated "Value" field MUST be populated by the entire header field being logged. That is, the field-name, the associated colon (":") and the field-value. This mechanism provides the capability to optionally log any SIP header field by identifying the field being logged within the "Value" field.

Because the Reason-Phrase in a response is part of the Status-Line and is not identified with a field-name, it is a special case. In this instance, the the associated "Value" field MUST be populated by the name "Reason-Phrase" followed by a colon (":") and a single space (SP) between the colon and the logged Reason-Phrase value.

The corresponding "Length" field includes the length of the entire "Value" field. This includes the field-name, the colon, and any LWS separator. For Tag=00 the BEB is set according to whether the SIP Header Field value contains any 'unprintable' characters. If it does not the BEB=00; if it does the BEB=01. If BEB=01 then only the field-value MUST be Base64 encoded; the field-name, the associated colon and any LWS separator MUST retain their original encoding.

If an optional field occurs more than once in a SIP message (e.g. Contact, Route, Record-Route, etc.) then each occurrence MUST be logged with the same Tag value (i.e., Tag=00) as a distinct optional field entry in the SIP CLF record. These repeated optionally logged header fields MUST preserve the ordinal position of the repeated header fields in the SIP header. For example, a SIP header containing two Via header fields with the following ordinal positions within the SIP header: V1,V2. If optionally logging these header fields they would occur as the following entries in the SIP CLF record. (Note: For the sake of brevity this example only shows how these optional header fields would be logged and omits the remainder of the SIP CLF record):

```
00@00000000,len_V1,00,Via: V1     00@00000000,len_V2,00,Via: V2
```

The terminating carriage return line feed (CRLF) after a given header field value MUST NOT be logged. Since a bare CRLF sequence is not permitted within a SIP header field value, optional SIP header fields logged with Tag=00 MUST NOT contain a CRLF when logged and consequently no escaping mechanism is required for it.

Tag = 01 - Log message body

SIP message bodies of all types can be optionally logged using Tag=01. If the message body is logged it MUST adhere to the maximum size limitation of 4096 bytes for a SIP CLF field, as detailed in Section 8 of [I-D.ietf-sipclf-problem-statement]. Unlike with Tag=00, there can only be a single entry in the SIP CLF record with Tag=01. When optionally logging the message body if the maximum SIP CLF field size of 4096 bytes is exceeded the message body being logged MUST be truncated to meet these size limitations.

When logging a message body (Tag=01), the associated "Value" field is populated with the Content-Type itself plus the SIP message body separated with a space. In this manner, everything about the SIP message body is self-described using a single tag as compared to enumerating a separate tag for each body type. Additionally, the corresponding "Length" field includes the SIP message body, the length of the embedded Content-Type, and the space separator between the MIME type and the body content.

For an optionally logged message body (Tag=01) the BEB is set according to whether the message body contains any 'unprintable' characters. If it does not the BEB=00; if it does the BEB=01. If BEB=01 then the message body that follows is entirely Base64 encoded except the prepended Content-Type as described in the previous paragraph.

If an optionally logged SIP message body contains any CRLFs they MUST be escaped by using the URI encoded equivalent value of "%0D%0A". This escaping mechanism applies to all body types. So we don't make any distinction in treatment between the various possible body types. If a logged message body has BEB=01 then it MUST be Base64 encoded prior to any character escaping. Thus, if a binary body (like an image) is logged it will be Base64 encoded first and that Base64 character stream could never include the CRLF escape sequence of "%0D%0A" because "%" is not a valid Base64 character.

Tag = 02 - Log entire SIP message

The entire SIP message (i.e., SIP header and message body) can be optionally logged using a Tag=02. Logging the entire SIP message MUST conform to the maximum size limitation of 4096 bytes for a SIP CLF field, as detailed in Section 8 of [I-D.ietf-sipclf-problem-statement]. Unlike with Tag=00, there can only be a single entry in the SIP CLF record with Tag=02. When optionally logging the entire SIP message if the maximum SIP CLF field size of 4096 bytes is exceeded the entire SIP message being logged MUST be truncated to meet these size limitations.

When optionally logging an entire SIP message (Tag=02) the BEB is set according to whether the message body portion contains any 'unprintable' characters. If it does not the BEB=00; if it does the BEB=01. If BEB=01 then the entire SIP message is Base64 encoded (not just the message body). Note that unlike the case of Tag=01, when logging an entire SIP message (Tag=02) with 'unprintable' characters (BEB=01) the Content-Type would

not be known prior to decode.

All instances of CRLFs, whether they appear in the SIP headers or the SIP message body MUST be escaped by using the URI encoded equivalent value of "%0D%0A". If a logged SIP message has BEB=01 then it MUST be Base64 encoded prior to any character escaping.

(2) Vendor-ID = PEN

A Vendor-ID set to a vendor's own private enterprise number from the complete current list of private enterprise numbers maintained by IANA [PEN] is used to log any other vendor-specified optional element of a SIP header or body. The value of the Tag is set at the discretion of the implementer:

Tag = Vendor-specified tag

The definition of the various values of the optional field identifier (Tag@Vendor-ID) are the basis of how optional elements are logged in the SIP CLF. For the sake of completeness the remaining fields in the format shown in Figure 5 are also defined below:

Length Field (4 bytes): Indicates the length of only the "Value" field of this optionally logged element (as shown in Figure 5), hexadecimal encoded. This length corresponds to the length of the "Value" field only and MUST NOT include any of the other elements shown in Figure 5.

Base64 Encoded Byte (BEB) Field (1 byte): Indicates whether the subsequent Value Field of the optionally logged element is Base64 encoded or not. The Value field for the optional element being logged MUST be Base64 encoded if it contains any character that is deemed 'unprintable' according to the definition given previously in this section. If the optional element being logged is Base64 encoded then BEB=0x01; if it is not Base64 encoded then BEB=0x00.

Value Field (0 to 4096 bytes): Contains the actual value of this optional field. As with the mandatory fields, UTF-8 Tab characters (0x09) are replaced with UTF-8 space characters (0x20).

The following are examples of optionally logged SIP elements using the syntax described in this section. All these examples only show the <OptionalFields> portion of the SIP CLF record. The mandatory <IndexPointers> and <MandatoryFields> portions of the SIP CLF are intentionally omitted for the sake of brevity. Note that all of these examples of optionally logged fields begin with a leading Tab delimiter byte (0x09) that is not apparent here.

(1) Contact header field logged as an optional field:

```
Consider the SIP response:

SIP/2.0 180 Ringing
<allOneLine>
Via: SIP/2.0/UDP host.example.com;
branch=z9hG4bKnashds8;received=192.0.2.1
</allOneLine>
To: Bob <sip:bob@example.com>;tag=a6c85cf
From: Alice <sip:alice@example.com>;tag=1928301774
Call-ID: a84b4c76e66710
Contact: <sip:bob@192.0.2.4>
CSeq: 314159 INVITE
Content-Length: 0
```

The Contact header field would be logged as an optional field in the following manner:

```
00@00000000,001C,00,Contact: <sip:bob@192.0.2.4>
```

(2) Reason-Phrase logged as an optional field:

For the same SIP response the Reason-Phrase would be logged as an optional field in the following manner:

```
00@00000000,0016,00,Reason-Phrase: Ringing
```

(3) SDP body to be logged as an optional field:

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.example.com
s=-
c=IN IP4 host.example.com
t=0 0
m=audio 49170 RTP/AVP 0 8 97
```

This body has a Content-Type of application/sdp and is of length of 123 bytes including all the line-feeds. When logging this

body the "Value" field is composed of the Content-Type and the body separated by a space, which gives it a combined length of 139 (0x008B) bytes. This SIP body would be logged as an optional field in the following manner:

```
 <allOneLine>
 01@00000000,008B,00,application/sdp v=0%0D%0Ao=alice 2890844526
 2890844526 IN IP4 host.example.com%0D%0As=-%0D%0A
 c=IN IP4 host.example.com%0D%0At=0 0%0D%0A
 m=audio 49170 RTP/AVP 0 8 97%0D%0A
 </allOneLine>
```

Note that the body is actually logged on a single line and is thus captured between <allOneLine/> tags. The line-feeds are escaped using %0D%0A to delimit the various lines in the message body.

(4) binary body to be logged as an optional field:

The second body part of the multipart/mime SIP message shown in Section 3.1.1.11 of RFC4475 is a binary encoded body (represented in hex) and if logged would have BEB=01 and would require Base64 encoding. That binary body would produce six lines of output after being Base64 encoded. Subsequent escaping of the CRLF characters would produce an optionally logged body that would look like:

```
<allOneLine>
01@00000000,0216,01,multipart/mixed;boundary=7a9cbec02ceef655 MI
IBUgYJKoZIhvcNAQcCoIIBQzCCAT8CAQExCTAHBgUrDgMCGjALBgkqhkiG9w0BBw
ExggEgMIIB%0D%0AHAIBATB8MHAxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxp
Zm9ybmlhMREwDwYDVQQHEwhTYW4g%0D%0ASm9zZTEOMAwGA1UEChMFc2lwaXQxKT
AnBgNVBAsTIFNpcGl0IFRlc3QgQ2VydGlmaWNhdGUgQXV0%0D%0AaG9yaXR5AggB
lQBxAjMBEzAHBgUrDgMCGjANBgkqhkiG9w0BAQEFAASBgI70ZvlI8FIt0uWXjp2V
%0D%0Aquny/hWgZllxYpLo2iqo2DUKaM7/rjy9K/8Wdd3VZI5ZPdZHKPJiIPfpQX
SeMw2aFe2r25PRDEIQ%0D%0ALntyidKcwMmuLvvHwM/5Fy87An5PwCfhVG3ktqo6
uz5mzMtd1sZLg4MUnLjm/xgtlE/le2W8mdAF%0D%0A
</allOneLine>
```

Note that the body is actually logged on a single line and is thus captured between <allOneLine/> tags. The line-feeds are escaped using %0D%0A to delimit the various lines in the Base64

encoded binary body.

(5) Codec information from the SDP body logged as an optional field:

Consider the SIP message:

```
INVITE sip:bob@example.com SIP/2.0
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bKnashds8
To: Bob <bob@example.com>
From: Alice <alice@example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@host.example.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=UserA 2890844526 2890844526 IN IP4 example.com
s=Session SDP
c=IN IP4 host.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

A vendor may choose to log a SIP message element such as the codec information from the SDP body. This Vendor-specified SIP element would be logged as an optional field in the following manner:

```
03@00032473,0014,00,a=rtpmap:0 PCMU/8000
```

(6) N-th message received from a particular peer logged as an optional field:

Perhaps a vendor wants to log that this message is the n-th message received from a peering partner. To do so for the SIP message shown above, the vendor would log this information as:

```
07@00032473,0016,00,1877 example.com
```

Which would signify that this is the 1,877th message from the peering partner example.com. Note that the previous two examples showing an optionally logged Vendor-specified SIP element use a Vendor-ID with a Private Enterprise Number of 32473. This value has been reserved by IANA to be used as an example PEN in documentation according to [RFC5612].

5. Example SIP CLF Record

The following SIP message is an INVITE request sent by a SIP client:

```
INVITE sip:192.0.2.10 SIP/2.0
To: <sip:192.0.2.10>
Call-ID: DL70dff590c1-1079051554@example.com
<allOneLine>
From: "Alice" <sip:1001@example.com:5060>;
tag=DL88360fa5fc;epid=0x34619b0
</allOneLine>
CSeq: 1 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2012 15:02:03 GMT
<allOneLine>
Via: SIP/2.0/TCP 192.0.2.200:5060;
branch=z9hG4bK-1f6be070c4-DL
</allOneLine>
Contact: "1001" <sip:1001@192.0.2.200:5060>
Content-Type: application/sdp
Content-Length: 418

v=0
o=1001 1456139204 0 IN IP4 192.0.2.200
s=Session SDP
c=IN IP4 192.0.2.200
b=AS:2048
t=0 0
m=audio 13756 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
```

Shown below is approximately how this message would appear as a single record in a SIP CLF logging file if encoded according to the syntax described in this document. Due to RFC conventions, this log entry has been split into five lines, instead of the two lines that actually appear in a log file; and the tab characters have been padded out using spaces to simulate their appearance in a text terminal.

```
       A000100,0053005C005E006D007D008F009E00A000BA00C700EB00F70100
       <allOneLine>
       1328821153.010    RORUU    1 INVITE    -    sip:192.0.2.10
       192.0.2.10:5060    192.0.2.200:56485    sip:192.0.2.10    -
       sip:1001@example.com:5060    DL88360fa5fc
       DL70dff590c1-1079051554@example.com    S1781761-88    C67651-11
       </allOneLine>
```

A bit-exact version of the actual log entry is provided here, Base64 encoded.

```
 begin-base64 644 clf_record
 QTAwMDEwMCwwMDUzMDA1QzAwNUUwMDZEMDA3RDAwOEYwMDlFMDBBMDAwQkEwMEM3MDBF
 QjAwRjcwMTAwCjEzMjg4MjExNTMuMDEwCVJPUlVVCTEgSU5WSVRFCS0Jc2lwOjE5Mi4w
 LjIuMTAJMTkyLjAuMi4xMDo1MDYwCTE5Mi4wLjIuMjAwOjU2NDg1CXNpcDoxOTIuMC4y
 LjEwCS0Jc2lwOjEwMDFAZXhhbXBsZS5jb206NTA2MAlETDg4MzYwZmE1ZmMJREw3MGRm
 ZjU5MGMxLTEwNzkwNTE1NTRAZXhhbXBsZS5jb20JUzE3ODE3NjEtODgJQzY3NjUxLTEx
 Cg==
 ====
```

To recover the unencoded file, the Base64 text above may be passed as input to the following perl script (the output should be redirected to a file).

```
<CODE BEGINS>

#!/usr/bin/perl
use strict;
my $bdata = "";
use MIME::Base64;
while(<>)
{
      if (/begin-base64 644 clf_record/ .. /-- ==== --/)
      {
          if ( m/^\s*[^\s]+\s*$/)
          {
            $bdata = $bdata . $_;
          }
       }
}
print decode_base64($bdata);

<CODE ENDS>
```

6. Text Tool Considerations

This format has been designed to allow text tools to easily process logs without needing to understand the indexing format. Index lines may be rapidly discarded by checking the first character of the line: index lines will always start with an alphabetical character, while field lines will start with a numerical character.

Within a field line, script tools can quickly split fields at the tab characters. The first 12 fields are positional, and the meaning of any subsequent fields can be determined by checking the first four characters of the field. Alternately, these non-positional fields can be located using a regular expression. For example, the "Contact value" in a request can be found by searching for the perl regex /\t0000,....,([^\t]*)/.

7. Security Considerations

This document does not introduce any new security considerations beyond those discussed in [I-D.ietf-sipclf-problem-statement].

In the interest of protecting the sensitive information contained in a SIP CLF file, [I-D.ietf-sipclf-problem-statement] notes that values might need to be obfuscated for privacy reasons when SIP CLF files are exchanged between domains. If a base64 encoded string contains the non-obfuscated value then that would also need to be obfuscated before base64 encoding.

8. Operational Guidance

SIP CLF log files will take up substantive amount of disk space depending on traffic volume at a processing entity and the amount of information being logged. As such, any enterprise using SIP CLF should establish operational procedures for file rollovers as appropriate to the needs of the organization.

Listing such operational guidelines in this document is out of scope for this work.

9. IANA Considerations

This specification establishes a new Session Initiation Protocol (SIP) Common Log Format (CLF) Parameters registry, which contains two new sub-registries for SIP CLF Version and SIP CLF Transport Flag. Initial entries are defined by this specification for both sub-

registries. Addition of any new sub-registry to the Session Initiation Protocol (SIP) Common Log Format (CLF) Parameters registry is to be done using the IETF Review registration policy detailed in [RFC5226].

9.1. SIP CLF Version

This document defines the SIP CLF "Version" field in Section 4.1. IANA has created a registry of Version values entitled "SIP CLF Version Values". Version numbers MUST be incremented for any new SIP CLF protocol specification that changes any part of the SIP CLF record format. Changes include addition or removal of fields or a change of syntax or semantics of existing fields.

Version numbers must be registered via the Standards Action method as described in [RFC5226]. IANA has registered the Versions shown in Table 1 below.

Version	FORMAT	Reference
0x41	Defined in RFCXXXX	RFCXXXX

Table 1: IANA-Registered SIP CLF Versions

[[NOTE TO RFC EDITOR: Please change XXXX to the number assigned to this specification, and remove this paragraph on publication.]]

9.2. SIP CLF Transport Flag

This document defines the SIP CLF "Transport Flag" as fourth byte in the Flags Field of the SIP CLF record. The format and values of the Transport Flag are described in Section 4.2. IANA has created a registry of SIP CLF Transport Flag values entitled "SIP CLF Transport Flag Values".

SIP CLF Transport Flag values must be registered via the IETF Review method as described in [RFC5226]. IANA has registered the Transport Flag values shown in Table 2 below.

Value	Transport Protocol	Reference
U	UDP	RFCXXXX
T	TCP	RFCXXXX
S	SCTP	RFCXXXX

Table 2: IANA-Registered SIP CLF Transport Flag

[[NOTE TO RFC EDITOR: Please change XXXX to the number assigned to this specification, and remove this paragraph on publication.]]

10. Acknowledgements

The authors of this document would like to acknowledge and thank Peter Musgrave (the chair of the SIPCLF working group) and Robert Sparks (the assigned area director) for their support, guidance, and continued invaluable feedback.

This work benefited from the discussions and invaluable input by the various members of the SIPCLF working group. These include Brian Trammell, Eric Burger, Cullen Jennings, Benoit Claise, Saverio Niccolini, Dan Burnett. Special thanks to Hadriel Kaplan, Chris Lonvick, Paul E. Jones, John Elwell, Claudio Allocchio, and Joe Clarke for their constructive comments, suggestions, and reviews that were critical to the formulation and refinement of this document.

Thanks to Anders Nygren for his early implementation, insight, and reviews of the SIP CLF format.

This document was written with the xml2rfc tool described in [RFC2629].

11. References

11.1. Normative References

[I-D.ietf-sipclf-problem-statement]
Gurbani, V., Burger, E., Anjali, T., Abdelnur, H., and O. Festor, "The Common Log Format (CLF) for the Session Initiation Protocol (SIP): Framework and Data Model", draft-ietf-sipclf-problem-statement-11 (work in progress), March 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

11.2. Informative References

[PEN] IANA, "Private Enterprise Numbers", http://www.iana.org/assignments/enterprise-numbers , 2009.

[RFC1166] Kirkpatrick, S., Stahl, M., and M. Recker, "Internet numbers", RFC 1166, July 1990.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC4475] Sparks, R., Hawrylyshen, A., Johnston, A., Rosenberg, J., and H. Schulzrinne, "Session Initiation Protocol (SIP) Torture Test Messages", RFC 4475, May 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, March 2009.

[RFC5612] Eronen, P. and D. Harrington, "Enterprise Number for Documentation Use", RFC 5612, August 2009.

[RFC5737] Arkko, J., Cotton, M., and L. Vegoda, "IPv4 Address Blocks Reserved for Documentation", RFC 5737, January 2010.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

[UNICODE] "The Unicode Consortium, The Unicode Standard, Version 6.2.0, (Mountain View, CA: ISBN 978-1-936213-07-8)", http://www.unicode.org/versions/Unicode6.2.0/ , 2012.

[W3C.REC-xml-20081126]
Sperberg-McQueen, C., Yergeau, F., Maler, E., Paoli, J., and T. Bray, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <http://www.w3.org/TR/2008/REC-xml-20081126>.

Authors' Addresses

Gonzalo Salgueiro
Cisco Systems
7200-12 Kit Creek Road
Research Triangle Park, NC 27709
US

Email: gsalguei@cisco.com

Vijay Gurbani
Bell Labs, Alcatel-Lucent
1960 Lucent Lane
Rm 9C-533
Naperville, IL 60563
US

Email: vkg@bell-labs.com

Adam Roach
Tekelec
17210 Campbell Rd.
Suite 250
Dallas, TX 75252
US

Email: adam@nostrum.com