Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on November 8, 2009.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (http://trustee.ietf.org/license-info). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

This document proposes a binary syntax for the SIP common log format (CLF). It does not cover semantic issues, and is meant to be evaluated in the context of the other efforts discussing SIP CLF.
1. Introduction

The Common Log File (CLF) format for the Session Initiation Protocol (SIP) [I-D.gurbani-sipping-clf] proposes a syntax for logging SIP messages received and sent by SIP clients, servers, and proxies. The syntax proposed by that document has been inspired by the common HTTP log format. However, experience with that format has shown that dealing with large quantities of log data can be very processor intensive, as doing so necessary requires reading and parsing every byte in the log file(s) of interest.

This document counter-proposes a format that is no more difficult to generate by logging entities, 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. Both operations can be performed in constant time (as compared with $O(n)$ time associated with the current format, where $n$ is the length of the log record).

Further, the format proposed by this document retains the ability to be read by humans and processed using traditional Unix text processing tools, such as sed, awk, perl, cut, and grep.

2. Format

Each data record is encoded according to the following format. Note that indications of "hexadecimal encoded" indicate that the value is to be written out in human-readable base-16 numbers using the ASCII characters 0x30 through 0x39 and 0x41 through 0x46 ('0' through '9' and '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 ASCII 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 ASCII '0' characters to fill the entire space.

```
<table>
<thead>
<tr>
<th>Version</th>
<th>Flags Field</th>
<th>0 - 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2C</td>
<td>Record Length</td>
<td>4 - 7</td>
</tr>
<tr>
<td></td>
<td>Record Length (cont)</td>
<td>8 - 11</td>
</tr>
<tr>
<td></td>
<td>Server Txn Pointer (Hex)</td>
<td>12 - 15</td>
</tr>
</tbody>
</table>
```
<table>
<thead>
<tr>
<th>Server Txn Length (Hex)</th>
<th>16 - 19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Txn Pointer (Hex)</td>
<td>20 - 23</td>
</tr>
<tr>
<td>Client Txn Length (Hex)</td>
<td>24 - 27</td>
</tr>
<tr>
<td>Method Pointer (Hex)</td>
<td>28 - 31</td>
</tr>
<tr>
<td>Method Length (Hex)</td>
<td>32 - 35</td>
</tr>
<tr>
<td>To Value Pointer (Hex)</td>
<td>36 - 39</td>
</tr>
<tr>
<td>To Value Length (Hex)</td>
<td>40 - 43</td>
</tr>
<tr>
<td>To Tag Pointer (Hex)</td>
<td>44 - 47</td>
</tr>
<tr>
<td>To Tag Length (Hex)</td>
<td>48 - 51</td>
</tr>
<tr>
<td>From Value Pointer (Hex)</td>
<td>52 - 55</td>
</tr>
<tr>
<td>From Value Length (Hex)</td>
<td>56 - 59</td>
</tr>
<tr>
<td>From Tag Pointer (Hex)</td>
<td>60 - 63</td>
</tr>
<tr>
<td>From Tag Length (Hex)</td>
<td>64 - 67</td>
</tr>
<tr>
<td>Call-Id Pointer (Hex)</td>
<td>68 - 71</td>
</tr>
<tr>
<td>Call-Id Length (Hex)</td>
<td>72 - 75</td>
</tr>
<tr>
<td>TLV Start Pointer (Hex)</td>
<td>76 - 79</td>
</tr>
<tr>
<td>0x0A</td>
<td>80 - 83</td>
</tr>
<tr>
<td>Date/Time</td>
<td>84 - 87</td>
</tr>
<tr>
<td>0x2E</td>
<td>88 - 91</td>
</tr>
<tr>
<td>Fractional Seconds</td>
<td>92 - 95</td>
</tr>
<tr>
<td>0x09</td>
<td>96 - 99</td>
</tr>
<tr>
<td>CSeq</td>
<td>100 - 103</td>
</tr>
<tr>
<td>104 - 107</td>
<td></td>
</tr>
<tr>
<td>0x09</td>
<td>Response</td>
</tr>
</tbody>
</table>
First, an 80-byte header indicates meta-data about the record. Note that the field lengths encoded in the header do not include the ASCII tab characters used to separate fields from each other.

Version (1 byte): 0x41 for this document

Flags Field (3 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 (r = message received, s = message sent)

Record Length (6 bytes): Hexadecimal-Encoded Total length of this log record, including "Flags" and "Record Length" fields, and terminating line-feed.

Bytes 12 through 72 contain hexadecimal-encoded pointer/length pairs that point to the values of variable-length mandatory fields. The "Pointer" fields indicate absolute byte values within the record, and must be >= 103. They point to the start of the corresponding value within the "Mandatory Fields" area. The "Length" fields indicate the length of the corresponding value. The final pointer, "TLV Start Pointer," points to the ASCII Tab (0x09) character for the first entry in the Tag/Length/Value area; if no such entries are present, this value is set to zero.

Note that the "Length" fields do not include the tab delimiters.
between fields. Further note that there are no delimiters between these pointer/length values -- they are packed together as a single, 68-character hexadecimal encoded string.

Following the pointer/length pairs, several fixed-length fields are encoded. As before, all fields are completely filled, pre-pending values with '0' characters as necessary.

Date/Time (10 bytes): Seconds since midnight, January 1st, 1970, GMT; decimal encoded

Fraction Seconds (6 bytes): Microseconds since the time in Date/Time field; decimal encoded

CSeq Number (10 bytes): CSeq number from the SIP message; decimal encoded

Response Code (3 bytes): Set to the value of the response code for responses. Set to 0 for requests. Decimal encoded.

Mandatory Field Data: Contains actual values for the mandatory fields. This data must appear in the order listed, and each field must be present. Fields are separated by a single ASCII Tab character (0x09). Any tab characters present in the data to be written will be replaced by an ASCII space character (0x20) prior to being logged.

Server Txn: The transaction identifier associated with the server transaction. Implementations MAY reuse the server transaction identifier (the topmost branch-id of the incoming request, with or without the magic cookie), or they MAY 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.

Client Txn: This field is used to associate client transactions with a server transaction for forking proxies or B2BUAs.

Method: In requests, the method from the start line. In responses, the method found in the CSeq header field.

To Value: Value of the To header field, possibly with the tag parameter removed. (Whether to remove the tag parameter is left up to the logging entity).
To Tag: Value of the To header field tag parameter. If no To header field tag parameter is present, the pointer field is ignored; the length field is set to 0; and the field in the mandatory section is encoded as a single ASCII dash (0x2D).

From Value: Value of the From header field, possibly with the tag parameter removed. (Whether to remove the tag parameter is left up to the logging entity)

From Tag: Value of the From header field tag parameter.

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

After the "Mandatory Fields" section, Tag/Length/Value groups appear zero or more times. The location within the log record is indicated by the "TLV Start Ptr" field. They are used to log information that is not mandatory for all messages (although specific TLVs are mandatory in request logs).

Tag Field (4 bytes): indicates the type of value coded by this TLV; hexadecimal encoded. Currently defined tags are:

0x0000 - Contact value (can be repeated)
Contains entire value of Contact header field

0x0001 - Request URI (mandatory in request)
Contains Request URI in start line

0x0002 - Remote Host (mandatory in request)
The DNS name of IP address from which the message was received (if "sent/received flag" is 0) or the IP address to which the message is being send (if "sent/received flag" is 1)

0x0003 - Authenticated User
Contains the user name by which the user has been authenticated

0x0004 - Complete SIP Message (optional, should be omitted by default)
Contains complete SIP message. Can be repeated multiple times to accommodate SIP messages that exceed 65535 bytes in length.
Length Field (2 bytes): indicates the length of the value coded in this TLV, hexadecimal encoded. This length does NOT include the TLV header.

Value Field (0 to 65535 bytes): contains the actual value of this TLV. As with the mandatory fields, ASCII Tab characters (0x09) are replaced with ASCII space characters (0x20).

3. Example Record

The following demonstrates approximately how a single log record appears in a logging file. Due to internet-draft conventions, this log entry has been split into ten 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.

ARor,000181, 00720000800000008100060088002000A9000600B0002500D6000A00E100270108
1241708241.308241 0000000187 000 7yuz67jhyi9-9 - INVITE Bob <sip:bob@biloxi.example.com> 314159
Alice <sip:alice@atlanta.example.com> 9fxced76s1 3848276298220188511@atlanta.example.com 0000,0034, <sip:alice@client.atlanta.example.com;transport=tcp>
0001,001A, sip:bob@biloxi.example.com 0002,000C,192.168.9.12

A uuencoded version of this log entry (without the changes required to format it for an internet-draft) follows.

begin 644 sip-clf.txt
M05)<BPP,#’Q.#$L,#’W,C’P,$OP,#’P,#’P,#’SP,#’V,#’X.#’P,C’P M,$$Y,#’P-C’P0C’P,#(U,#!$-C’P,$$P,$4Q,#’R-S’Q,#@*,3(T,3<P.#(T M,2XS,#&R-#$),#’P,#’P,#$X-PDP,#’)-WEU>C8W:FAY:3DM.0DM"4E.5DE4 M10E;V(8/-I#ib;v) ’8FEL;WAI+F5X86UP;&4N8V)]M/0DS,30Q-3D06QI M8V4B/-I#IA;&EC94!A=&QA;G1A+F5X86UP;&4N8V]M/0DY9GAC960W-G-L M"3,X-#R-S8R.30R,C’Q.#0U,3’871L86YT82YE>&%<&QE+F-0;0DP,#’P M+#’P,SOL/-I#IA;EC94!C;EE;GON871L86YT82YE>&%<&QE+F-0;3MT M<&%<X;W!O<9G]=-&P/0DP,#’Q+#’P,4$L<VEP,F)O8D!B:6QO<&DN97AA;7!L =92YC;VT),#’P,BPP,#!+##Y,BXQ-C0N.2XQ,#H`
end
4. 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 11 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 "Request URI" in a request can be found by searching for the perl regex /	0001,....,([\^t]*)/.

Note also that requests can be distinguished from responses by checking the third positional field -- for requests, it will always be set to "000"; any other value indicates a response.

5. Normative References

[I-D.gurbani-sipping-clf]

Appendix A. Acknowledgements

Cullen put me up to this.

Tom Taylor suggested the technique of combining the length field structure from the binary format with the human-readable ASCII format to allow both rapid processing by advanced tools, and easy processing by simpler, text-centric tools. Dean Willis suggested the use of tab delimiters as a means to avoid the need to escape values within a field. Vijay Gurbani provided significant feedback, and wrote the original proof-of-concept program which was adapted to produce the examples in this document.
Author’s Address

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

Email: adam@nostrum.com