SIPCLF
Internet-Draft
Intended status: Standards Track
Expires: August 9, 2012

V. Gurbani, Ed.
Bell Laboratories, Alcatel-Lucent
E. Burger, Ed.
Georgetown University
T. Anjali
Illinois Institute of Technology
H. Abdelnur
O. Festor
INRIA

February 6, 2012

The Common Log Format (CLF) for the Session Initiation Protocol (SIP): Framework and Data Model
draft-ietf-sipclf-problem-statement-10

Abstract

Well-known web servers such as Apache and web proxies like Squid support event logging using a common log format. The logs produced using these de-facto standard formats are invaluable to system administrators for trouble-shooting a server and tool writers to craft tools that mine the log files and produce reports and trends. Furthermore, these log files can also be used to train anomaly detection systems and feed events into a security event management system. The Session Initiation Protocol (SIP) does not have a common log format, and as a result, each server supports a distinct log format that makes it unnecessarily complex to produce tools to do trend analysis and security detection. We propose a common log file format for SIP servers that can be used uniformly by user agents, proxies, registrars, redirect servers as well as back-to-back user agents.

Status of this Memo

This Internet-Draft will expire on August 9, 2012.

Table of Contents

1. 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 RFC 2119 [RFC2119].

RFC 3261 [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.

2. Introduction

Servers executing on Internet hosts produce log records as part of their normal operations. Some log records are, in essence, a summary of an application layer protocol data unit (PDU), that captures in precise terms an event that was processed by the server. These log records serve many purposes, including analysis and troubleshooting.

Well-known web servers such as Apache and Squid support event logging using a Common Log Format (CLF), the common structure for logging requests and responses serviced by the web server. It can be argued that a good part of the success of Apache has been its CLF because it allowed third parties to produce tools that analyzed the data and generated traffic reports and trends. The Apache CLF has been so successful that not only did it become the de-facto standard in producing logging data for web servers, but also many commercial web servers can be configured to produce logs in this format. An example of Apache CLF is depicted next:

```
          %h      %l     %u       %t    \"%r\"   %s    %b
     remotehost rfc931 authuser [date] request status bytes
```

remotehost: Remote hostname (or IP number if DNS hostname is not available, or if DNSLookup is Off.

rfc931: The remote logname of the user.

authuser: The username by which the user has authenticated himself.

[date]: Date and time of the request.

request: The request line exactly as it came from the client.

status: The HTTP status code returned to the client.

bytes: The content-length of the document transferred.

The inspiration for the SIP CLF is the Apache CLF. However, the state machinery for a HTTP transaction is much simpler than that of the SIP transaction (as evidenced in Section 7). The SIP CLF needs to do considerably more.

This document outlines the problem statement that argues for a SIP CLF. In addition, it provides a data model pertaining to the minimum set of SIP headers and fields that must be logged. This document does not prescribe a specific representation format for the SIP CLF record and instead, allows other documents to define a representation format. [I-D.ietf-sipclf-format] is an example of a representation format that provides an UTF-8 based logging scheme.

3. Problem statement

The Session Initiation Protocol (SIP) [RFC3261] is an Internet multimedia session signaling protocol. A typical deployment of SIP in an enterprise will consist of SIP entities from multiple vendors. Currently, if these entities are capable of producing a log file of the transactions being handled by them, the log files are produced in a proprietary format. The result of multiplicity of the log file formats is the inability of the support staff to easily trace a call from one entity to another, or even to craft common tools that will perform trend analysis, debugging and troubleshooting problems uniformly across the SIP entities of multiple vendors.

SIP does not currently have a common log format and this document serves to provide the rationale to establish a SIP CLF and identifies the required minimal information that must appear in any SIP CLF record.

4. What SIP CLF is and what it is not

The SIP CLF is a standardized manner of producing a log file. This

format can be used by SIP clients, SIP Servers, proxies, and B2BUAs. The SIP CLF is simply an easily digestible log of currently occurring events and past transactions. It contains enough information to allow humans and automata to derive relationships between discrete transactions handled at a SIP entity or to search for a certain dialog or a related set of transactions.

The SIP CLF is amenable to quick parsing (i.e., well-delimited fields) and it is platform and operating system neutral.

The SIP CLF is amenable to easy parsing and lends itself well to creating other innovative tools.

The SIP CLF is not a billing tool. It is not expected that enterprises will bill customers based on SIP CLF. The SIP CLF records events at the signaling layer only and does not attempt to correlate the veracity of these events with the media layer. Thus, it cannot be used to trigger customer billing.

The SIP CLF is not a quality of service (QoS) measurement tool. If QoS is defined as measuring the mean opinion score (MOS) of the received media, then SIP CLF does not aid in this task since it does not summarize events at the media layer.

And finally, the SIP CLF is not a tool for supporting lawful intercept.

5. Alternative approaches to SIP CLF

It is perhaps tempting to consider other approaches --- which though not standardized, are in wide enough use in networks today --- to determine whether or not a SIP CLF would benefit a SIP network consisting of multi-vendor products. The two existing approaches that approximate what SIP CLF does are Call Detail Records (CDRs) and Wireshark packet sniffing.

5.1. SIP CLF and CDRs

CDRs are used in operator networks widely and with the adoption of SIP, standardization bodies such as 3GPP have subsequently defined SIP-related CDRs as well. Today, CDRs are used to implement the functionality approximated by SIP CLF, however, there are important differences.

One, SIP CLF operates natively at the transaction layer and maintains enough information in the information elements being logged that dialog-related data can be subsequently derived from the transaction

logs. Thus, esoteric SIP fields and parameters like the To header, including tags; the From header, including tags, the CSeq number, etc. are logged in SIP CLF. By contrast, a CDR is used mostly for charging and thus saves information to facilitate that very aspect. A CDR will most certainly log the public user identification of a party requesting a service (which may not correspond to the From header) and the public user identification of the party called party (which may not correspond to the To header.) Furthermore, the sequence numbers maintained by the CDR may not correspond to the SIP CSeq header. Thus it will be hard to piece together the state of a dialog through a sequence of CDR records.

Two, a CDR record will, in all probability, be generated at a SIP entity performing some form of proxy-like functionality of a B2BUA providing some service. By contrast, SIP CLF is light- weight enough that it can be generated by a canonical SIP user agent server and user agent client as well, including those that execute on resource constrained devices (mobile phones).

Finally, SIP is also being deployed outside of operator- managed VoIP networks. Universities, research laboratories, and small-to-medium size companies are deploying SIP-based VoIP solutions on networks owned and managed by them. Much of the latter constituencies will not have an interest in generating CDRs, but they will like to have a concise representation of the messages being handled by the SIP entities in a common format.

5.2. SIP CLF and Wireshark packet capture

Wireshark is a popular raw packet capture tool. It contains filters that can understand SIP at the protocol level and break down a captured message into its individual header components. While Wireshark is appropriate to capture and view discrete SIP messages, it does not suffice to serve in the same capacity as SIP CLF for the following reasons:

- Using Wireshark will not eliminate the need for agreeing to a common set of fields to represent a SIP CLF record. This common understanding is important for interoperability to allow one implementation to read a log file written by a different implementation.
- Using Wireshark would require that the underlying libraries related to Wireshark and packet capture be available for all platforms on which a SIP server or a SIP client can execute on. Given the different platforms that a SIP client or server runs on --- mobile, fixed host, tablet, etc. --- this may become an inhibiting factor when compared to the SIP client or server producing a SIP CLF record natively (the SIP client or server has

already parsed the SIP message for operation on it, therefore, it seems reasonable to have it write the parsed tokens out to persistent store in an agreed upon format).

- o If SIP messages are exchanged over a secure transport (TLS), Wireshark will be unable to decrypt them and render them as individual SIP headers.
- o Using Wireshark and the related packet capture libraries may imposes a dependency on a third party library.

6. Motivation and use cases

As SIP becomes pervasive in multiple business domains and ubiquitous in academic and research environments, it is beneficial to establish a CLF for the following reasons:

Common reference for interpreting events: In a laboratory environment or an enterprise service offering there will typically be SIP entities from multiple vendors participating in routing requests. Absent a common log format, each entity will produce output records in a native format making it hard to establish commonality for tools that operate on the log file.

Writing common tools: A common log format allows independent tool providers to craft tools and applications that interpret the CLF data to produce insightful trend analysis and detailed traffic reports. The format should be such that it retains the ability to be read by humans and processed using traditional Unix text processing tools.

Session correlation across diverse processing elements: In operational SIP networks, a request will typically be processed by more than one SIP server. A SIP CLF will allow the network operator to trace the progression of the request (or a set of requests) as they traverse through the different servers to establish a concise diagnostic trail of a SIP session.

Note that tracing the request through a set of servers is considerably less challenging if all the servers belong to the same administrative domain.

Message correlation across transactions: A SIP CLF can enable a quick lookup of all messages that comprise a transaction (e.g., "Find all messages corresponding to server transaction X, including all forked branches.")

Message correlation across dialogs: A SIP CLF can correlate transactions that comprise a dialog (e.g., "Find all messages for dialog created by Call-ID C, From tag F and To tag T.")

Trend analysis: A SIP CLF allows an administrator to collect data and spot patterns or trends in the information (e.g., "What is the domain where the most sessions are routed to between 9:00 AM and 1:00 PM?")

Train anomaly detection systems: A SIP CLF will allow for the training of anomaly detection systems that once trained can monitor the CLF file to trigger an alarm on the subsequent deviations from accepted patterns in the data set. Currently, anomaly detection systems monitor the network and parse raw packets that comprise a SIP message -- a process that is unsuitable for anomaly detection systems [rieck2008]. With all the necessary event data at their disposal, network operations managers and information technology operation managers are in a much better position to correlate, aggregate, and prioritize log data to maintain situational awareness.

Testing: A SIP CLF allows for automatic testing of SIP equipment by writing tools that can parse a SIP CLF file to ensure behavior of a device under test.

Troubleshooting: A SIP CLF can enable cursory trouble shooting of a SIP entity (e.g., "How long did it take to generate a final response for the INVITE associated with Call-ID X?")

Offline analysis: A SIP CLF allows for offline analysis of the data gathered. Once a SIP CLF file has been generated, it can be transported (subject to the security considerations in Section 10) to a host with appropriate computing resources to perform subsequent analysis.

Real-time monitoring: A SIP CLF allows administrators to visually notice the events occurring at a SIP entity in real-time providing accurate situational awareness.

7. Challenges in establishing a SIP CLF

Establishing a CLF for SIP is a challenging task. The behavior of a SIP entity is more complex when compared to the equivalent HTTP entity.

Base protocol services such as parallel or serial forking elicit

multiple final responses. Ensuing delays between sending a request and receiving a final response all add complexity when considering what fields should comprise a CLF and in what manner. Furthermore, unlike HTTP, SIP groups multiple discrete transactions into a dialog, and these transactions may arrive at a varying inter-arrival rate at a proxy. For example, the BYE transaction usually arrives much after the corresponding INVITE transaction was received, serviced and expunged from the transaction list. Nonetheless, it is advantageous to relate these transactions such that automata or a human monitoring the log file can construct a set consisting of related transactions.

ACK requests in SIP need careful consideration as well. In SIP, an ACK is a special method that is associated with an INVITE only. It does not require a response, and furthermore, if it is acknowledging a non-2xx response, then the ACK is considered part of the original INVITE transaction. If it is acknowledging a 2xx-class response, then the ACK is a separate transaction consisting of a request only (i.e., there is not a response for an ACK request.) CANCEL is another method that is tied to an INVITE transaction, but unlike ACK, the CANCEL request elicits a final response.

While most requests elicit a response immediately, the INVITE request in SIP can remain in a pending state at a proxy as it forks branches downstream or at a user agent server while it alerts the user. [RFC3261] instructs the server transaction to send a 1xx-class provisional response if a final response is delayed for more than 200 ms. A SIP CLF log file needs to include such provisional responses because they help train automata associated with anomaly detection systems and provide some positive feedback for a human observer monitoring the log file.

Finally, beyond supporting native SIP actors such as proxies, registrars, redirect servers, and user agent servers (UAS), it is beneficial to derive a common log format that supports back-to-back user agent (B2BUA) behavior, which may vary considerably depending on the specific nature of the B2BUA.

8. Data model

The minimal SIP CLF fields are defined below. Some of these fields contain URIs [RFC3986]. If the URI contains an escaped character (""%" HEX HEX" mechanism), the escaped character MUST be logged as received. The maximum size (in number of bytes) for a SIP CLF field is 4096 bytes. This limit is the same regardless of whether the SIP CLF field is a meta-field (see "Timestamp" and "Directionality" defined below) or a normal SIP header. If the body of the SIP message is to be logged, it MUST conform to this limit as well.

Logging bodies of a SIP message is left optional (and is not shown in the examples of Section 9). If the body is to be logged, the specific syntax and semantics used to log bodies MUST be defined by the specific representation format used to generate the SIP CLF record.

The data model supports extensibility by providing the capability to log "optional fields". Optional fields are those SIP header fields (or field components) that are not mandatory (see Section 8.1 for the mandatory field list). Optional fields may contain SIP headers or other elements present in a SIP message (for example, the Reason-Phrase element from the Status-Line production rule in RFC 3261 [RFC3261]). Optional fields may also contain additional information that a particular vendor desires to log. The specific syntax and semantics to be accorded to optional fields MUST be defined by the specific representation format used to generate the SIP CLF record.

8.1. SIP CLF mandatory fields

The following SIP CLF fields are defined as minimal information that MUST appear in any SIP CLF record:

Timestamp - Date and time of the request or response represented as the number of seconds and milliseconds since the Unix epoch.

Message type - An indicator on whether the SIP message is a request or a response. The allowable values for this field are 'R' (for Request) and 'r' (for response).

Directionality - An indicator on whether the SIP message is received by the SIP entity or sent by the SIP entity. The allowable values for this field are 's' (for message sent) and 'r' (for message received).

Transport - The transport over which a SIP message is sent or received. The allowable values for the transport are governed by the "transport" production rule in Section 25.1 of RFC3261 [RFC3261].

Source-address - The IPv4 or IPv6 address of the sender of the SIP message.

Source-port - The source port number of the sender of the SIP message.

Destination-address - The IPv4 or IPv6 address of the recipient of the SIP message.

Destination-port - The port number of the recipient of the SIP message.

From - The From URI. For the sake of brevity, URI parameters SHOULD NOT be logged.

From-tag - The tag parameter of the From header.

To - The To URI. For the sake of brevity, URI parameters SHOULD NOT be logged.

To-tag - The tag parameter of the To header. Note that the tag parameter will be absent in the initial request that forms a dialog.

Callid - The Call-ID.

CSeq-Method - The method from the CSeq header.

CSeq-Number - The number from the CSeq header.

R-URI - The Request-URI, including any URI parameters.

Status - The SIP response status code.

SIP Proxies may fork, creating several client transactions that correlate to a single server transaction. Responses arriving on these client transactions, or new requests (CANCEL, ACK) sent on the client transaction need log file entries that correlate with a server transaction. Similarly, a B2BUA may create one or more client transactions in response to an incoming request. These transactions will require correlation as well. The last two data model elements provide this correlation.

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 as explained later in this section. (See Section 9 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 for usage.)

This data model applies to all SIP entities --- a UAC, UAS, Proxy, a B2BUA, registrar and redirect server. The SIP CLF fields prescribed for a proxy are equally applicable to the B2BUA. Similarly, the SIP CLF fields prescribed for a UAS are equally applicable to registrars and redirect servers.

The next section specifies the individual SIP CLF data model elements that form a log record for specific instance of a SIP entity. It is understood that a SIP CLF record is extensible using extension mechanisms appropriate to the specific representation used to generate the SIP CLF record. This document, however, does not prescribe a specific representation format and it limits the discussion to the mandatory data elements described above.

8.2. Mandatory fields and SIP entities

Each SIP CLF record MUST consist of all the mandatory data model elements outlined in Section 8.1. This document does not specify a representation of a logging format; it is expected that other documents will do so. Each SIP CLF record MUST contain the mandatory elements shown below:

    Timestamp, Message type, Directionality, CSeq-Method,
    CSeq-Number, Transport, R-URI, Destination-address,
    Destination-port, Source-address, Source-port, To,
    To-tag, From, From-tag, Call-ID, Status, Server-Txn,
    Client-Txn

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. Therefore, the representation document MUST define how to indicate a field is "not applicable". For example, the R-URI field is not applicable when logging a response, the Status field is not applicable when logging a request, the To-tag is not known when a request is first sent out, etc.

The Client-Txn field is always applicable to a UAC. The Server- Txn field does not apply to a UAC unless the element is also acting as a UAS, and the message associated to this log record corresponds to a message handled by that UAS. For instance, a proxy forwarding a

request will populate both the Client-Txn and Server-Txn fields in the record corresponding to the forwarded request.

The Server-Txn field is always applicable to a UAS. The Client-Txn field does not apply to a UAS unless the element is also acting as a UAC, and the message associated to this log record corresponds to a message handled by that UAC. For instance, a proxy forwarding a response will populate both the Server-Txn and Client-Txn fields in the record corresponding to the forwarded response. However, a proxy would only populate the Client-Txn field when creating a log record corresponding to a request.

9. Examples

The examples use only the mandatory data elements defined in Section 8.1. Extension elements are not considered. When a given mandatory field is not applicable to a SIP entity, we use the horizontal dash ("-") to represent it.

There are five principals in the examples below. They are Alice, the initiator of requests. Alice's user agent uses IPv4 address 198.51.100.1, port 5060. P1 is a proxy that Alice's request traverse on their way to Bob, the recipient of the requests. P1 also acts as a registrar to Alice. P1 uses an IPv4 address of 198.51.100.10, port 5060. Bob has two instances of his user agent running on different hosts. The first instance uses an IPv4 address of 203.0.113.1, port 5060 and the second instance uses an IPv6 address of 2001:db8::9, port 5060. P2 is a proxy responsible for Bob's domain. Table 1 summarizes these addresses.

Principal	IP:port	Host/Domain name
Alice	198.51.100.1:5060	alice.example.com
P1	198.51.100.10:5060	p1.example.com
P2	203.0.113.200:5060	p2.example.net
Bob UA instance 1	203.0.113.1:5060	bob1.example.net
Bob UA instance 2	[2001:db8::9]:5060	bob2.example.net

Principal to IP address asignment

Table 1

Illustrative examples of SIP CLF follow.

9.1. UAC registration

Alice sends a registration registrar P1 and receives a 2xx-class response. The register requests causes Alice's UAC to produce a log record shown below.

```
Timestamp: 1275930743.699
Message Type: R
Directionality: s
Transport: udp
CSeq-Number: 1
CSeq-Method: REGISTER
R-URI: sip:example.com
Destination-address: 198.51.100.10
Destination-port: 5060
Source-address: 198.51.100.1
Source-port: 5060
To: sip:example.com
To-tag: -
From: sip:alice@example.com
From-tag: 76yhh
Call-ID: f81-d4-f6@example.com
Status: -
Server-Txn: -
Client-Txn: c-tr-1
```

After some time, Alice's UAC will receive a response from the registrar. The response causes Alice's agent to produce a log record shown below.

```
Timestamp: 1275930744.100
Message Type: r
Directionality: r
Transport: udp
CSeq-Number: 1
CSeq-Method: REGISTER
R-URI: -
Destination-address: 198.51.100.1
Destination-port: 5060
Source-address: 198.51.100.10
Source-port: 5060
To: sip:example.com
To-tag: reg-1-xtr
From: sip:alice@example.com
From-tag: 76yhh
Call-ID: f81-d4-f6@example.com
Status: 100
Server-Txn: -
```

```
Client-Txn: c-tr-1
```

9.2. Direct call between Alice and Bob

In this example, Alice sends a session initiation request directly to Bob's agent (instance 1.) Bob's agent accepts the session invitation. We first present the SIP CLF logging from Alice's UAC point of view. In line 1, Alice's user agent sends out the INVITE. Shortly, it receives a "180 Ringing" (line 2), followed by a "200 OK" response (line 3). Upon the receipt of the 2xx-class response, Alice's user agent sends out an ACK request (line 4).

```
Timestamp: 1275930743.699
Message Type: R
Directionality: s
Transport: udp
CSeq-Number: 32
CSeq-Method: INVITE
R-URI: sip:bob@bob1.example.net
Destination-address: 203.0.113.1
Destination-port: 5060
Source-address: 198.51.100.1
Source-port: 5060
To: sip:bob@bob1.example.net
To-tag: -
From: sip:alice@example.com
From-tag: 76yhh
Call-ID: f82-d4-f7@example.com
Status: -
Server-Txn: -
Client-Txn: c-1-xt6

Timestamp: 1275930745.002
Message Type: r
Directionality: r
Transport: udp
CSeq-Number: 32
CSeq-Method: INVITE
R-URI: -
Destination-address: 198.51.100.1
Destination-port: 5060
Source-address: 203.0.113.1
Source-port: 5060
To: sip:bob@example.net
To-tag: b-in6-iu
From: sip:alice@example.com
From-tag: 76yhh
Call-ID: f82-d4-f7@example.com
```

```
Status: 180
Server-Txn: -
Client-Txn: c-1-xt6

Timestamp: 1275930746.100
Message Type: r
Directionality: r
Transport: udp
CSeq-Number: 32
CSeq-Method: INVITE
R-URI: -
Destination-address: 198.51.100.1
Destination-port: 5060
Source-address: 203.0.113.1
Source-port: 5060
To: sip:bob@example.net
To-tag: b-in6-iu
From: sip:alice@example.com
From-tag: 76yhh
Call-ID: f82-d4-f7@example.com
Status: 200
Server-Txn: -
Client-Txn: c-1-xt6

Timestamp: 1275930746.120
Message Type: R
Directionality: s
Transport: udp
CSeq-Number: 32
CSeq-Method: ACK
R-URI: sip:bob@bob1.example.net
Destination-address: 203.0.113.1
Destination-port: 5060
Source-address: 198.51.100.1
Source-port: 5060
To: sip:bob@example.net
To-tag: b-in6-iu
From: sip:alice@example.com
From-tag: 76yhh
Call-ID: f82-d4-f7@example.com
Status: -
Server-Txn: -
Client-Txn: c-1-xt6
```

9.3. Single downstream branch call

In this example, Alice sends a session invitation request to Bob through proxy P1, which inserts a Record-Route header causing subsequent requests between Alice and Bob to traverse the proxy. The SIP CLF log records correspond to the viewpoint of P1. The line numbers below refer to Figure 1

```
     Alice             P1             Bob
      +---INV--------->|               |  Line 1
      |                |               |
      |<---------100---+               |  Line 2
      |                |               |
      |                +---INV-------->|  Line 3
      |                |               |
      |                |<--------100---+  Line 4
      |                |               |
      |                |<--------180---+  Line 5
      |                |               |
      |<---------180---+               |  Line 6
      |                |               |
      |                |<--------200---+  Line 7
      |                |               |
      |<---------200---+               |  Line 8
      |                |               |
      +---ACK--------->|               |  Line 9
      |                |               |
      |                |---ACK-------->|  Line 10
```

Figure 1: Simple proxy-aided call flow

```
1    Timestamp: 1275930743.699
     Message Type: R
     Directionality: r
     Transport: udp
     CSeq-Number: 43
     CSeq-Method: INVITE
     R-URI: sip:bob@example.net
     Destination-address: 198.51.100.10
     Destination-port: 5060
     Source-address: 198.51.100.1
     Source-port: 5060
     To: sip:bob@example.net
     To-tag: -
     From: sip:alice@example.com
     From-tag: al-1
     Call-ID: tr-87h@example.com
     Status: -
     Server-Txn: s-x-tr
     Client-Txn: -
```

Note that at this point P1 has created a server transaction identification code and populated the SIP CLF field Server-Txn with it. P1 has not yet created a client transaction identification code, thus Client-Txn contains a "-".

```
2    Timestamp: 1275930744.001
     Message Type: r
     Directionality: s
     Transport: udp
     CSeq-Number: 43
     CSeq-Method: INVITE
     R-URI: -
     Destination-address: 198.51.100.1
     Destination-port: 5060
     Source-address: 198.51.100.10
     Source-port: 5060
     To: sip:bob@example.net
     To-tag: -
     From: sip:alice@example.com
     From-tag: al-1
     Call-ID: tr-87h@example.com
     Status: 100
     Server-Txn: s-x-tr
     Client-Txn: -
```

In line 3 below, P1 has created a client transaction identification

code for the downstream branch and populated the SIP CLF field
Client-Txn.

```
3    Timestamp: 1275930744.998
     Message Type: R
     Directionality: s
     Transport: udp
     CSeq-Number: 43
     CSeq-Method: INVITE
     R-URI: sip:bob@bob1.example.net
     Destination-address: 203.0.113.1
     Destination-port: 5060
     Source-address: 198.51.100.10
     Source-port: 5060
     To: sip:bob@example.net
     To-tag: -
     From: sip:alice@example.com
     From-tag: al-1
     Call-ID: tr-87h@example.com
     Status: -
     Server-Txn: s-x-tr
     Client-Txn: c-x-tr

4    Timestamp: 1275930745.200
     Message Type: r
     Directionality: r
     Transport: udp
     CSeq-Number: 43
     CSeq-Method: INVITE
     R-URI: -
     Destination-address: 198.51.100.10
     Destination-port: 5060
     Source-address: 203.0.113.1
     Source-port: 5060
     To: sip:bob@example.net
     To-tag: b1-1
     From: sip:alice@example.com
     From-tag: al-1
     Call-ID: tr-87h@example.com
     Status: 100
     Server-Txn: s-x-tr
     Client-Txn: c-x-tr

5    Timestamp: 1275930745.800
     Message Type: r
     Directionality: r
     Transport: udp
     CSeq-Number: 43
```

```
       CSeq-Method: INVITE
       R-URI: -
       Destination-address: 198.51.100.10
       Destination-port: 5060
       Source-address: 203.0.113.1
       Source-port: 5060
       To: sip:bob@example.net
       To-tag: b1-1
       From: sip:alice@example.com
       From-tag: al-1
       Call-ID: tr-87h@example.com
       Status: 180
       Server-Txn: s-x-tr
       Client-Txn: c-x-tr

  6    Timestamp: 1275930746.009
       Message Type: r
       Directionality: s
       Transport: udp
       CSeq-Number: 43
       CSeq-Method: INVITE
       R-URI: -
       Destination-address: 198.51.100.1
       Destination-port: 5060
       Source-address: 198.51.100.10
       Source-port: 5060
       To: sip:bob@example.net
       To-tag: b1-1
       From: sip:alice@example.com
       From-tag: al-1
       Call-ID: tr-87h@example.com
       Status: 180
       Server-Txn: s-x-tr
       Client-Txn: c-x-tr

  7    Timestamp: 1275930747.120
       Message Type: r
       Directionality: r
       Transport: udp
       CSeq-Number: 43
       CSeq-Method: INVITE
       R-URI: -
       Destination-address: 198.51.100.10
       Destination-port: 5060
       Source-address: 203.0.113.1
       Source-port: 5060
       To: sip:bob@example.net
       To-tag: b1-1
```

```
      From: sip:alice@example.com
      From-tag: al-1
      Call-ID: tr-87h@example.com
      Status: 200
      Server-Txn: s-x-tr
      Client-Txn: c-x-tr

   8  Timestamp: 1275930747.300
      Message Type: r
      Directionality: s
      Transport: udp
      CSeq-Number: 43
      CSeq-Method: INVITE
      R-URI: -
      Destination-address: 198.51.100.1
      Destination-port: 5060
      Source-address: 198.51.100.10
      Source-port: 5060
      To: sip:bob@example.net
      To-tag: b1-1
      From: sip:alice@example.com
      From-tag: al-1
      Call-ID: tr-87h@example.com
      Status: 200
      Server-Txn: s-x-tr
      Client-Txn: c-x-tr

   9  Timestamp: 1275930749.100
      Message Type: R
      Directionality: r
      Transport: udp
      CSeq-Number: 43
      CSeq-Method: ACK
      R-URI: sip:bob@example.net
      Destination-address: 198.51.100.10
      Destination-port: 5060
      Source-address: 198.51.100.1
      Source-port: 5060
      To: sip:bob@example.net
      To-tag: b1-1
      From: sip:alice@example.com
      From-tag: al-1
      Call-ID: tr-87h@example.com
      Status: -
      Server-Txn: s-x-tr
      Client-Txn: c-x-tr

   10 Timestamp: 1275930749.100
```

```
Message Type: R
Directionality: s
Transport: udp
CSeq-Number: 43
CSeq-Method: ACK
R-URI: sip:bob@bob1.example.net
Destination-address: 203.0.113.1
Destination-port: 5060
Source-address: 198.51.100.10
Source-port: 5060
To: sip:bob@example.net
To-tag: b1-1
From: sip:alice@example.com
From-tag: al-1
Call-ID: tr-87h@example.com
Status: -
Server-Txn: s-x-tr
Client-Txn: c-x-tr
```

9.4. Forked call

In this example, Alice sends a session invitation to Bob's proxy, P2. P2 forks the session invitation request to two registered endpoints corresponding to Bob's address-of-record. Both endpoints respond with provisional responses. Shortly thereafter, one of Bob's user agent instances accepts the call, causing P2 to send a CANCEL request to the second user agent. P2 does not Record-Route, therefore the subsequent ACK request from Alice to Bob's user agent does not traverse through P2 (and is not shown below.)

Figure 2 depicts the call flow.

```
                           Bob            Bob
        Alice      P2   (Instance 1) (Instance 2)
         +---INV--->|          |         |  Line 1
         |          |          |         |
         |<---100---+          |         |  Line 2
         |          |          |         |
         |          +---INV--->|         |  Line 3
         |          |          |         |
         |          +---INV----+-------->|  Line 4
         |          |          |         |
         |          |<---100---+         |  Line 5
         |          |          |         |
         |          |<---------+---100---+  Line 6
         |          |          |         |
         |          |<---180---+---------+  Line 7
         |          |          |         |
         |<---180---+          |         |  Line 8
         |          |          |         |
         |          |<---180---+         |  Line 9
         |          |          |         |
         |<---180---+          |         |  Line 10
         |          |          |         |
         |          |<---200---+         |  Line 11
         |          |          |         |
         |<---200---+          |         |  Line 12
         |          |          |         |
         |          +---CANCEL-+-------->|  Line 13
         |          |          |         |
         |          |<---------+---487---+  Line 14
         |          |          |         |
         |          +---ACK----+-------->|  Line 15
         |          |          |         |
         |          |<---------+---200---+  Line 16
```

Figure 2: Forked call flow

The SIP CLF log correspond to the viewpoint of P2. The fields logged are shown below; the line numbers refer to Figure 2.

```
1    Timestamp: 1275930743.699
     Message Type: R
     Directionality: r
     Transport: udp
     CSeq-Number: 43
     CSeq-Method: INVITE
     R-URI: sip:bob@example.net
```

```
      Destination-address: 203.0.113.200
      Destination-port: 5060
      Source-address: 198.51.100.1
      Source-port: 5060
      To: sip:bob@example.net
      To-tag: -
      From: sip:alice@example.com
      From-tag: a1-1
      Call-ID: tr-88h@example.com
      Status: -
      Server-Txn: s-1-tr
      Client-Txn: -

   2  Timestamp: 1275930744.001
      Message Type: r
      Directionality: s
      Transport: udp
      CSeq-Number: 43
      CSeq-Method: INVITE
      R-URI: -
      Destination-address: 198.51.100.1
      Destination-port: 5060
      Source-address: 203.0.113.200
      Source-port: 5060
      To: sip:bob@example.net
      To-tag: -
      From: sip:alice@example.com
      From-tag: a1-1
      Call-ID: tr-88h@example.com
      Status: 100
      Server-Txn: s-1-tr
      Client-Txn: -

   3  Timestamp: 1275930744.998
      Message Type: R
      Directionality: s
      Transport: udp
      CSeq-Number: 43
      CSeq-Method: INVITE
      R-URI: sip:bob@bob1.example.net
      Destination-address: 203.0.113.1
      Destination-port: 5060
      Source-address: 203.0.113.200
      Source-port: 5060
      To: sip:bob@example.net
      To-tag: -
      From: sip:alice@example.com
      From-tag: a1-1
```

```
        Call-ID: tr-88h@example.com
        Status: -
        Server-Txn: s-1-tr
        Client-Txn: c-1-tr

   4    Timestamp: 1275930745.500
        Message Type: R
        Directionality: s
        Transport: udp
        CSeq-Number: 43
        CSeq-Method: INVITE
        R-URI: sip:bob@bob2.example.net
        Destination-address: [2001:db8::9]
        Destination-port: 5060
        Source-address: 203.0.113.200
        Source-port: 5060
        To: sip:bob@example.net
        To-tag: -
        From: sip:alice@example.com
        From-tag: a1-1
        Call-ID: tr-88h@example.com
        Status: -
        Server-Txn: s-1-tr
        Client-Txn: c-2-tr

   5    Timestamp: 1275930745.800
        Message Type: r
        Directionality: r
        Transport: udp
        CSeq-Number: 43
        CSeq-Method: INVITE
        R-URI: -
        Destination-address: 203.0.113.200
        Destination-port: 5060
        Source-address: 203.0.113.1
        Source-port: 5060
        To: sip:bob@example.net
        To-tag: b1=-1
        From: sip:alice@example.com
        From-tag: a1-1
        Call-ID: tr-88h@example.com 100
        Status: 100
        Server-Txn: s-1-tr
        Client-Txn: c-1-tr

   6    Timestamp: 1275930746.100
        Message Type: r
        Directionality: r
```

```
        Transport: udp
        CSeq-Number: 43
        CSeq-Method: INVITE
        R-URI: -
        Destination-address: 203.0.113.200
        Destination-port: udp
        Source-address: [2001:db8::9]
        Source-port: 5060
        To: sip:bob@example.net
        To-tag: b2-2
        From: sip:alice@example.com
        From-tag: a1-1
        Call-ID: tr-88h@example.com
        Status: 100
        Server-Txn: s-1-tr
        Client-Txn: c-2-tr

   7    Timestamp: 1275930746.700
        Message Type: r
        Directionality: r
        Transport: udp
        CSeq-Number: 43
        CSeq-Method: INVITE
        R-URI: -
        Destination-address: 203.0.113.200
        Destination-port: udp
        Source-address: [2001:db8::9]
        Source-port: 5060
        To: sip:bob@example.net
        To-tag: b2-2
        From: sip:alice@example.com
        From-tag: a1-1
        Call-ID: tr-88h@example.com
        Status: 180
        Server-Txn: s-1-tr
        Client-Txn: c-2-tr

   8    Timestamp: 1275930746.990
        Message Type: r
        Directionality: s
        Transport: udp
        CSeq-Number: 43
        CSeq-Method: INVITE
        R-URI: -
        Destination-address: 198.51.100.1
        Destination-port: 5060
        Source-address: 203.0.113.200
        Source-port: 5060
```

```
       To: sip:bob@example.net
       To-tag: b2-2
       From: sip:alice@example.com
       From-tag: a1-1
       Call-ID: tr-88h@example.com
       Status: 180
       Server-Txn: s-1-tr
       Client-Txn: c-2-tr

   9   Timestamp: 1275930747.100
       Message Type: r
       Directionality: r
       Transport: udp
       CSeq-Number: 43
       CSeq-Method: INVITE
       R-URI: -
       Destination-address: 203.0.113.200
       Destination-port: 5060
       Source-address: 203.0.113.1
       Source-port: 5060
       To: sip:bob@example.net
       To-tag: b1-1
       From: sip:alice@example.com
       From-tag: a1-1
       Call-ID: tr-88h@example.com 100
       Status: 180
       Server-Txn: s-1-tr
       Client-Txn: c-1-tr

   10  Timestamp: 1275930747.300
       Message Type: r
       Directionality: s
       Transport: udp
       CSeq-Number: 43
       CSeq-Method: INVITE
       R-URI: -
       Destination-address: 198.51.100.1
       Destination-port: 5060
       Source-address: 203.0.113.200
       Source-port: 5060
       To: sip:bob@example.net
       To-tag: b1-1
       From: sip:alice@example.com
       From-tag: a1-1
       Call-ID: tr-88h@example.com
       Status: 180
       Server-Txn: s-1-tr
       Client-Txn: c-2-tr
```

```
   11   Timestamp: 1275930747.800
        Message Type: r
        Directionality: r
        Transport: udp
        CSeq-Number: 43
        CSeq-Method: INVITE
        R-URI: -
        Destination-address: 203.0.113.200
        Destination-port: 5060
        Source-address: 203.0.113.1
        Source-port: 5060
        To: sip:bob@example.net
        To-tag: b1-1
        From: sip:alice@example.com
        From-tag: a1-1
        Call-ID: tr-88h@example.com 100
        Status: 200
        Server-Txn: s-1-tr
        Client-Txn: c-1-tr

   12   Timestamp: 1275930748.000
        Message Type: r
        Directionality: s
        Transport: udp
        CSeq-Number: 43
        CSeq-Method: INVITE
        R-URI: -
        Destination-address: 198.51.100.1
        Destination-port: 5060
        Source-address: 203.0.113.200
        Source-port: 5060
        To: sip:bob@example.net
        To-tag: b1-1
        From: sip:alice@example.com
        From-tag: a1-1
        Call-ID: tr-88h@example.com
        Status: 200
        Server-Txn: s-1-tr
        Client-Txn: c-1-tr

   13   Timestamp: 1275930748.201
        Message Type: R
        Directionality: s
        Transport: udp
        CSeq-Number: 43
        CSeq-Method: CANCEL
        R-URI: sip:bob@bob2.example.net
        Destination-address: [2001:db8::9]
```

```
        Destination-port: 5060
        Source-address: 203.0.113.200
        Source-port: 5060
        To: sip:bob@example.net
        To-tag: b2-2
        From: sip:alice@example.com
        From-tag: a1-1
        Call-ID: tr-88h@example.com
        Status: -
        Server-Txn: s-1-tr
        Client-Txn: c-2-tr

   14   Timestamp: 1275930748.300
        Message Type: r
        Directionality: r
        Transport: udp
        CSeq-Number: 43
        CSeq-Method: INVITE
        R-URI: -
        Destination-address: 203.0.113.200
        Destination-port: udp
        Source-address: [2001:db8::9]
        Source-port: 5060
        To: sip:bob@example.net
        To-tag: b2-2
        From: sip:alice@example.com
        From-tag: a1-1
        Call-ID: tr-88h@example.com
        Status: 487
        Server-Txn: s-1-tr
        Client-Txn: c-2-tr

   15   Timestamp: 1275930748.355
        Message Type: R
        Directionality: s
        Transport: udp
        CSeq-Number: 43
        CSeq-Method: ACK
        R-URI: sip:bob@bob2.example.net
        Destination-address: [2001:db8::9]
        Destination-port: 5060
        Source-address: 203.0.113.200
        Source-port: 5060
        To: sip:bob@example.net
        To-tag: b2-2
        From: sip:alice@example.com
        From-tag: a1-1
        Call-ID: tr-88h@example.com
```

```
      Status: -
      Server-Txn: s-1-tr
      Client-Txn: c-2-tr

   16 Timestamp: 1275930748.698
      Message Type: r
      Directionality: r
      Transport: udp
      CSeq-Number: 43
      CSeq-Method: CANCEL
      R-URI: -
      Destination-address: 203.0.113.200
      Destination-port: udp
      Source-address: [2001:db8::9]
      Source-port: 5060
      To: sip:bob@example.net
      To-tag: b2-2
      From: sip:alice@example.com
      From-tag: a1-1
      Call-ID: tr-88h@example.com
      Status: 200
      Server-Txn: s-1-tr
      Client-Txn: c-2-tr
```

The above SIP CLF log makes it easy to search for a specific transaction or a state of the session. Searching for the string "c-1-tr" on the log records will readily yield the information that an INVITE was sent to sip:bob@bob1.example.com, it elicited a 100 followed by a 180 and then a 200. Because the ACK request in this case would be exchanged end-to-end, this element does not see (and therefore will not log) the ACK.

Searching on "c-2-tr" yields a more complex scenario of sending an INVITE to sip:bob@bob2.example.net, receiving 100 and 180. However, the log makes it apparent that the request to sip:bob@bob2.example.net was subsequently CANCEL'ed before a final response was generated, and that the pending INVITE returned a 487. The ACK to the final non-2xx response and a 200 to the CANCEL request complete the exchange on that branch.

10. Security Considerations

A log file by its nature reveals both the state of the entity producing it and the nature of the information being logged. To the extent that this state should not be publicly accessible and that the information is to be considered private, appropriate file and directory permissions attached to the log file SHOULD be used. It is

outside the scope of this document to specify how to protect the log file while it is stored on disk, however, certain precautions can be taken. Operators SHOULD consider using common administrative features such as disk encryption and securing log files [schneier-1]. Operators SHOULD also consider hardening the machine on which the log file is stored by restricting physical access to the host as well as restricting access to the file itself. Depending on the specific operating system and environment, the file and directory permissions SHOULD be set to be most restrictive such that the file is not publicly readable and writable and the directory where the file is stored is not publicly accessible.

The following threats may be considered for the log file while it is stored:

- An attacker may gain access to view the log file, or may surreptitiously make a copy of the log file for later viewing.
- An attacker who is unable to eavesdrop real-time SIP traffic on the network but nonetheless can access the log file, is able to easily mount replay attack or other attacks that result from channel eavesdropping. Encrypting SIP traffic does not help here because the SIP entity generating the log file would have decrypted the message for processing and subsequent logging.
- An attacker may delete parts of --- or indeed, the whole --- file.

Public access to the SIP log file creates more of a privacy leak when compared to an adversary eavesdropping cleartext SIP traffic on the network. If all SIP traffic on a network segment is encrypted, then as noted above, special attention must be directed to the file and directory permissions associated with the log file to preserve privacy such that only a privileged user can access the contents of the log file.

Transporting SIP CLF files across the network pose special challenges as well. The following threats may be considered for transferring log files or while transferring individual log records:

- An attacker may view the records;
- An attacker may modify the records in transit or insert previously captured records into the stream;
- An attacker may remove records in transit, or may stage a man- in-the-middle attack to deliver a partially or entirely falsified log file.

It is also outside the scope of this document to specify protection methods for log files or log records that are being transferred between hosts, however, certain precautions can be taken. Operators SHOULD require mutual authentication, channel confidentiality and

channel integrity while transferring the log file. The use of a secure shell transport layer protocol [RFC4253] or TLS [RFC5246] accomplishes this.

Even with such care, sensitive information can be leaked during or after the transfer. SIP CLF fields like IP addresses and URIs contain potentially sensitive information. Before transferring the log file across domains, operators SHOULD ensure that any fields that contain sensitive information are appropriately anonymized or obfuscated.

The SIP CLF represents the minimum fields that lend themselves to trend analysis and serve as information that may be deemed useful. Other formats can be defined that include more headers (and the body) from Section 8.1. However, where to draw a judicial line regarding the inclusion of non-mandatory headers can be challenging. Clearly, the more information a SIP entity logs, the longer time the logging process will take, the more disk space the log entry will consume, and the more potentially sensitive information could be breached. Therefore, adequate tradeoffs should be taken in account when logging more fields than the ones recommended in Section 8.1.

Implementers need to pay particular attention to buffer handling when reading or writing log files. SIP CLF entries can be unbounded in length. It would be reasonable for a full dump of a SIP message to be thousands of octets long. This is of particular importance to CLF log parsers, as a SIP CLF log writers may add one or more extension fields to the message to be logged.

11. 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 organization 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.

12. IANA Considerations

This document does not require any considerations from IANA.

13. Acknowledgments

Members of the sipping, dispatch, ipfix and syslog working groups provided invaluable input to the formulation of the draft. These include Benoit Claise, Spencer Dawkins, John Elwell, David Harrington, Christer Holmberg, Hadriel Kaplan, Atsushi Kobayashi, Jiri Kuthan, Scott Lawrence, Chris Lonvick, Peter Musgrave, Simon Perreault, Adam Roach, Dan Romascanu, Robert Sparks, Brian Trammell, Dale Worley, Theo Zourzouvillys and others that we have undoubtedly, but inadvertently, missed.

Rainer Gerhards, David Harrington, Cullen Jennings and Gonzalo Salgueiro helped tremendously in discussions related to arriving at the beginnings of a data model.

14. References

14.1. Normative References

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

14.2. Informative References

[I-D.ietf-sipclf-format]
Salgueiro, G., Gurbani, V., and A. Roach, "Format for the Session Initiation Protocol (SIP) Common Log Format (CLF)", draft-ietf-sipclf-format-05 (work in progress), December 2011.

[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.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4253] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[rieck2008]
Rieck, K., Wahl, S., Laskov, P., Domschitz, P., and K-R.

Muller, "A Self-learning System for Detection of Anomalous SIP Messages", Principles, Systems and Applications of IP Telecommunications Services and Security for Next Generation Networks (IPTComm), LNCS 5310, pp. 90-106, 2008.

[schneier-1]
Schneier, B. and J. Kelsey, "Secure audit logs to support computer forensics", ACM Transactions on Information and System Security (TISSEC), 2(2), pp. 159,176, May 1999.

Authors' Addresses

Vijay K. Gurbani (editor)
Bell Laboratories, Alcatel-Lucent
1960 Lucent Lane
Naperville, IL 60566
USA

Email: vkg@bell-labs.com

Eric W. Burger (editor)
Georgetown University
USA

Email: eburger@standardstrack.com
URI: http://www.standardstrack.com

Tricha Anjali
Illinois Institute of Technology
316 Siegel Hall
Chicago, IL 60616
USA

Email: tricha@ece.iit.edu

Humberto Abdelnur
INRIA
INRIA - Nancy Grant Est
Campus Scientifique
54506, Vandoeuvre-les-Nancy Cedex
France

Email: humbol@gmail.com

Olivier Festor
INRIA
INRIA - Nancy Grant Est
Campus Scientifique
54506, Vandoeuvre-les-Nancy Cedex
France

Email: Olivier.Festor@loria.fr