

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: April 28, 2011

S. Niccolini  
NEC  
B. Claise  
Cisco Systems Inc.  
B. Trammell  
ETH Zurich  
H. Kaplan  
Acme Packet  
October 25, 2010

A Common Log Format for SIP using IPFIX Files  
draft-niccolini-sipclf-ipfix-05

Abstract

This draft defines a log file format conforming to the information model defined in the SIP-CLF problem statement based upon the IPFIX File Format. It details the creation and interpretation of these files, and provides examples based on common SIP situations.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on April 28, 2011.

Copyright Notice

Copyright (c) 2010 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 (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. Introduction to IPFIX . . . . .	3
1.2. Encoding and Limits . . . . .	4
2. Information Elements for SIPCLF . . . . .	5
3. Recommended Templates for SIPCLF . . . . .	9
4. IPFIX File Format for SIPCLF . . . . .	11
4.1. Logging Raw SIP Messages . . . . .	11
4.2. Supporting fast searches of SIPCLF log files . . . . .	13
5. Efficiency Analysis . . . . .	13
6. Examples . . . . .	13
6.1. Base Template Export . . . . .	14
6.2. UAC registration . . . . .	14
6.3. Direct Call . . . . .	16
6.4. Single Downstream Branch Call . . . . .	17
6.5. Forked Call . . . . .	19
6.6. Torture Tests . . . . .	23
7. Security Considerations . . . . .	25
8. IANA Considerations . . . . .	25
9. Acknowledgments . . . . .	26
10. References . . . . .	26
10.1. Normative References . . . . .	26
10.2. Informative References . . . . .	26
Appendix A. Example messages in base64 . . . . .	27
Authors' Addresses . . . . .	29

## 1. Introduction

The SIPCLF WG is chartered to produce a format suitable for logging at any SIP element. The charter explicitly addresses the need to search, merge and summarize log records from one or more possibly diverse elements as well as the need to correlate messages from multiple elements. An additional consideration to be taken into account when defining the file format is the extensibility of SIP. SIPCLF is additionally chartered to identify the fields to appear in a log record and provide one or more formats for encoding those fields. As the IPFIX File format [RFC5655] meets these requirements, SIPCLF is presently considering an IPFIX File-based binary logging format, in addition to an ASCII text-based format. This document describes the IPFIX format.

Specifically, this document defines new Information Elements for the SIPCLF IPFIX file format (based on the data model detailed in [I-D.ietf-sipclf-problem-statement]), proposes common IPFIX Templates for this data model, uses these templates to define a format, and presents examples of an IPFIX File logging format for SIPCLF. For this purpose examples were taken from [I-D.ietf-sipclf-problem-statement], and torture tests to demonstrate the applicability of the format to uncommon situations from [RFC4475].

### 1.1. Introduction to IPFIX

IPFIX (IP Flow Export Protocol) [RFC5101] is an IETF Proposed Standard protocol for the export of network traffic information. In addition to a transport-agnostic unidirectional export protocol, it defines a simple encapsulation into files [RFC5655] which expands its applicability into logging and file storage. Originally designed for flows, its flexible and extensible data model lends itself readily to any situation involving events occurring on a network.

The central unit of storage in IPFIX is the Message. An IPFIX Message is composed of a header and one or more Sets. The header contains metadata about the message such as its length and when it was exported; see section 3.1 of [RFC5101] for details. Each Set is prefixed by a header containing an ID, which identifies the type of the set's contents; and the length of the set; this is detailed in section 3.3 of [RFC5101].

The ID of the set refers to a Template describing each Record within the Set. A Template is an ordered list of Information Elements drawn from a registry covering many fields relevant to network traffic export. New Information Elements may be added to the IANA registry to cover new applications, and may also be defined on the fly as

enterprise-specific Information Elements, scoped to private registries within an SMI Private Enterprise Number.

Templates are exported inline, alongside data, within IPFIX Sets identified by special Set IDs. In this way, an IPFIX Message stream is self-describing with minimal overhead in situations where the number of records is much greater than the number of record types. This is a feature of the SIPCLF data model, and indicates the applicability of IPFIX to SIPCLF logging.

## 1.2. Encoding and Limits

Each Information Element specifies a data type, covering most primitives (e.g. signed and unsigned integers, floats, booleans, strings) as well as some network-specific types (addresses, timestamps). All types other than strings and octet arrays are stored in network byte order. Within a Template, an Information Element may be specified to have a fixed length or a variable length; if the latter, the field in the Record has a length prefix of one or three bytes. One-byte prefixes represent lengths from 0 to 254; three byte prefixes are encoded as 255 followed by an unsigned16 in network byte order and can represent lengths from 0 to 65535. Typically, strings are stored with variable length, so IPFIX string fields can be thought of as length-prefixed (or "Pascal") strings. Two advantages of this encoding are that each variable-length field in IPFIX can be skipped with one or two comparisons (depending on the length representation), and that the framing itself is immune to misformatted strings (e.g., with embedded NULs).

Although IPFIX variable length Information Elements can represent single values up to 65535 bytes long, there is an additional length limit imposed by the Message format itself. Each Message has an unsigned 16-bit length in its header; this length includes the length of the header itself as well as any set headers, to facilitate easy framing and deframing of messages. Since the header is 16 bytes, and the set header 4 bytes, the minimum per-message overhead (a message with one set containing only one record) is 20 bytes, leading to a record limit of 65515 bytes. A record containing a single variable-length element with 3-byte length encoding therefore has a content limit of 65512 bytes.

In the common case, this limit has no practical impact. Each field within a SIPCLF message is normally limited to 4096 bytes as described in Section 2; 15 such full-length fields will fit in an IPFIX Message. However, when using IPFIX to export extended SIPCLF information, for example dumping full SIP body content during debugging, care needs to be taken; this is discussed further in Section 4.1.

## 2. Information Elements for SIPCLF

This section formally defines information elements for SIPCLF based on the data model proposed in [I-D.ietf-sipclf-problem-statement].

The IANA IPFIX Information Elements registry available at <http://www.iana.org/assignments/ipfix/ipfix.xhtml> defines a number of Information Elements useful to SIPCLF. These concern traditional flow key and timing information, and appear in the table below:

Name	Number	<Type>/Description
observationTimeSeconds	322	<dateTimeSeconds> Timestamp of the request or response represented as the number of seconds since the Unix epoch. Used when second-level precision is adequate.
observationTimeMilli-seconds	323	<dateTimeMilliseconds> Timestamp of the request or response represented as the number of milliseconds since the Unix epoch. Used when millisecond-level precision is required.
sourceIPv4Address	8	<ipv4Address> The IPv4 address of the source of the SIP message
sourceIPv6Address	27	<ipv6Address> The IPv6 address of the source of the SIP message
sourceTransportPort	7	<unsigned16> The source port number of source of the SIP message
destinationIPv4Address	12	<ipv4Address> The IPv4 address of the destination of the SIP message
destinationIPv6Address	28	<ipv6Address> The IPv6 address of the destination of the SIP message
destinationTransportPort	11	<unsigned16> The destination port number for the SIP message
protocolIdentifier	4	<unsigned8> The type of transport for the SIP message (UDP vs. TCP vs. SCTP)

In addition, we define the following Information Elements to represent the SIP-specific mandatory fields defined in [I-D.ietf-sipclf-problem-statement], many themselves taken from [RFC3261]. As these are not yet registered with IANA, but required to support experimental implementation during the development of this representation, these are provisionally located within the number space assigned to Private Enterprise Number (PEN), here registered within the private enterprise number 35566, which belongs to one of the authors of this draft. It is intended that these Information Elements be registered within the IANA number space, should this draft become a Proposed Standard.

Name	PEN/Number	<Type>/Description
sipObservationType	35566/419	<unsigned8> The observation type of this log entry. Denotes whether the entry was corresponds to a SIP message received, sent, or merely seen by a passive observer, as per the sipObservationType subregistry defined below.
sipMethod	35566/402	<unsigned8> The SIP method from the CSeq header, as per the sipMethod subregistry defined below.
sipFromURI	35566/404	<string> The From URI
sipFromTag	35566/405	<string> The From header field tag parameter value
sipToURI	35566/406	<string> The To URI.
sipToTag	35566/407	<string> The To header field tag parameter value, if present
sipCallId	35566/408	<string> The Call-ID header field value
sipSequenceNumber	35566/409	<unsigned32> The sequence number from the CSeq header.
sipRequestURI	35566/403	<string> The Request URI, including any parameters.
sipResponseStatus	35566/412	<unsigned16> The SIP response code returned upstream. The presence of this Information Element marks a record as describing a SIP response.
sipServerTransaction	35566/413	<string> The transaction identifier associated with the server transaction

sipClientTransaction	35566/414	<string> The transaction identifier associated with the server transaction
----------------------	-----------	--

The sipObservationType IE encodes whether the device logging the entry was the sender or received of the message. It can take the following four values:

Number	Name	Description
0	unknown	The logging process does not specify the observation type.
1	receiver	This entry was logged by the receiver of the message.
2	sender	This entry was logged by the sender of the message.
3	passive	This entry was logged by a passive observer of the message, e.g. a passive metering process parsing the SIP message from a sniffed packet stream.

The sipMethod IE encodes the supported methods from [RFC3261] as integers, by the order in which they appear in the IANA SIP Parameters Methods registry at <http://www.iana.org/assignments/sip-parameters>:

Number	Method	Notes
0	Unknown	The logging process did not recognize the SIP method.
1	ACK	defined in RFC3261
2	BYE	defined in RFC3261
3	CANCEL	defined in RFC3261
4	INFO	defined in RFC2976
5	INVITE	defined in RFC3261
6	MESSAGE	defined in RFC3428
7	NOTIFY	defined in RFC3265
8	OPTIONS	defined in RFC3261
9	PRACK	defined in RFC3262
10	PUBLISH	defined in RFC3903
11	REFER	defined in RFC3515
12	REGISTER	defined in RFC3261
13	SUBSCRIBE	defined in RFC3265

14	UPDATE	defined in RFC3311	
----	--------	--------------------	--

The following Information Elements are defined to represent additional SIP fields which have been identified as potentially useful by the SIPCLF working group.

Name	PEN/Number	<Type>/Description
sipAuthUsername	35566/401	<string> The SIP user name by which the user has been authenticated
sipContactURI	35566/410	<string> The Contact URI, possibly multiple
sipPaiURI	35566/411	<string> The P-Asserted-Identity URI
sipSessionId	35566/415	<octetArray> A 128-bit session identifier, as in [I-D.kaplan-dispatch-session-id]
sipMessageSection	35566/416	<octetArray> Section of a SIP body. If a sipMessageSectionOffset is present, this section begins that many octets into the SIP body; otherwise, the section begins with the start of the SIP body.
sipMessageSectionOffset	35566/417	<unsigned64> Offset of the original SIP body, in octets, from which the sipMessageSection in this record was taken.
sipMessageLength	35566/418	<unsigned64> Total length of the SIP body.

Any of the SIP-defined string Information Elements MAY be truncated by the logging process to support SIPCLF limits; if truncation is supported, the limits SHOULD be user-configurable, and the maximum limit SHOULD be 4096 bytes per field.

Bringing this all together, the additional Information Elements required by SIPCLF, in textual IE specification format [I-D.trammell-ipfix-text-iespec] is as follows; IE numbers appear in parentheses, IPFIX types in angle brackets, and sizes ("v" for variable-length) in square brackets:



```
sipAuthUsername(35566/401)<string>[v]
sipMethod(35566/402)<unsigned8>[1]
sipRequestURI(35566/403)<string>[v]
sipFromURI(35566/404)<string>[v]
sipFromTag(35566/405)<string>[v]
sipToURI(35566/406)<string>[v]
sipToTag(35566/407)<string>[v]
sipCallId(35566/408)<string>[v]
sipSequenceNumber(35566/409)<unsigned32>[4]
sipContactURI(35566/410)<string>[v]
sipPaiURI(35566/411)<string>[v]
sipResponseStatus(35566/412)<unsigned16>[2]
sipServerTransaction(35566/413)<string>[v]
sipClientTransaction(35566/414)<string>[v]
sipSessionId(35566/415)<octetArray>[16]
sipMessageSection(35566/416)<octetArray>[v]
sipMessageSectionOffset(35566/417)<unsigned64>[8]
sipMessageLength(35566/418)<unsigned64>[8]
sipObservationType(35566/419)<unsigned8>[8]
```

Figure 1: Information Model

### 3. Recommended Templates for SIPCLF

The following Templates are defined as recommended Templates for SIPCLF request and response messages. These are base templates, containing only mandatory Information Elements. Request and Response Templates MUST contain at least the Information Elements defined here. Optional Information Elements MAY be added to them, and the IPv4 addresses within these Templates MUST be replaced with IPv6 addresses for logging IPv6 transport of SIP messages. A sipServerTransaction Information Element SHOULD be added for all messages logged by a User Agent Server, and a sipClientTransaction Information Element SHOULD be added for all messages logged by a User Agent Client. These templates follow the recommended fields for request and response logging in [I-D.ietf-sipclf-problem-statement], and are defined using the representation in [I-D.trammell-ipfix-text-iespec].

```
observationTimeMilliseconds(323) [8]
sipSequenceNumber(35566/409) [4]
sourceIPv4Address(8) [4]
destinationIPv4Address(12) [4]
sourceTransportPort(7) [2]
destinationTransportPort(11) [2]
protocolIdentifier(4) [1]
sipMethod(35566/402) [1]
sipObservationType(35566/419) [1]
sipRequestURI(35566/403) [v]
sipToURI(35566/406) [v]
sipToTag(35566/407) [v]
sipFromURI(35566/404) [v]
sipFromTag(35566/405) [v]
sipCallId(35566/408) [v]
```

Figure 2: Base Request Template (IPv4)

```
observationTimeMilliseconds(323) [8]
sipSequenceNumber(35566/409) [4]
sourceIPv4Address(8) [4]
destinationIPv4Address(12) [4]
sourceTransportPort(7) [2]
destinationTransportPort(11) [2]
protocolIdentifier(4) [1]
sipMethod(35566/402) [1]
sipObservationType(35566/419) [1]
sipResponseStatus(35566/412) [2]
sipToURI(35566/406) [v]
sipToTag(35566/407) [v]
sipFromURI(35566/404) [v]
sipFromTag(35566/405) [v]
sipCallId(35566/408) [v]
```

Figure 3: Base Response Template (IPv4)

Note that the Information Elements in these templates are ordered to place the fixed-length elements before the variable-length ones, which speeds random access to fixed-length elements. However, since element order within a record is unimportant in IPFIX, any ordering of the mandatory Information Elements within a record **MUST** be accepted as a valid SIPCLF record for that record type.

The record type is determined by the presence of the `sipResponseStatus` field. If present in the Template, the Template describes a response record. If absent, it describes a request record.

#### 4. IPFIX File Format for SIPCLF

Now that we have defined Information Elements to support the SIPCLF Data Model and recommended Templates to define the records, we combine this data model to the IPFIX File format to define the SIPCLF format over IPFIX.

The IPFIX File format is defined in [RFC5655] as a serialized set of IPFIX Messages containing Data Records organized in Sets defined by Templates; these are in turn defined in the IPFIX Protocol specification [RFC5101]. The file format is designed to facilitate interoperability, flexibility, and reusability among a wide variety of storage, processing, and analysis tools.

A SIPCLF-IPFIX file is then defined as an IPFIX File as in [RFC5655] containing Templates containing at least the Information Elements defined in the recommended Templates above, followed by Data Sets containing records defined by those Templates.

The Template mechanism provided by IPFIX allows some flexibility here. A SIPCLF-IPFIX file writer MAY include other Information Elements in the Templates it uses for SIP logging (for example, to add additional information such as in the list of optional Information Elements, above), and MAY include Templates and records described by them which do not contain SIPCLF information; these latter records MAY be ignored by SIPCLF file readers.

Some specific considerations for SIPCLF-IPFIX files are detailed in the subsections below.

##### 4.1. Logging Raw SIP Messages

As noted in Section 1.2, the IPFIX Message format places a practical limit of 65515 bytes of content per Message. The base templates are structured so that this limit has no practical effect. However, SIPCLF Information Elements defined in this document MAY also be used to dump raw header and body information, and these may be longer than 65515 bytes. There are two ways to address this.

First, as with all other elements, the raw message can be truncated. This specification provides the sipMessageSection and sipMessageLength Information Elements to represent this. Here, the first N bytes of the message would be exported in the variable-length sipMessageSection Information Element, and the total length of the SIP body in the sipMessageLength Information Element. A template for requests received at a UAS is shown in Figure 4.

```
observationTimeMilliseconds(323) [8]
sipSequenceNumber(35566/409) [4]
sipMessageLength(35566/418) [4]
sourceIPv4Address(8) [4]
destinationIPv4Address(12) [4]
sourceTransportPort(7) [2]
destinationTransportPort(11) [2]
protocolIdentifier(4) [1]
sipMethod(35566/402) [1]
sipObservationType(35566/419) [1]
sipRequestURI(35566/403) [v]
sipToURI(35566/406) [v]
sipToTag(35566/407) [v]
sipFromURI(35566/404) [v]
sipFromTag(35566/405) [v]
sipCallId(35566/408) [v]
sipServerTransaction(35566/413) [v]
sipMessageSection(35566/416) [v]
```

Figure 4: Template for Request Record with Partial Raw Message at UAS

For exporting large raw SIP messages, a second continuation template is necessary, as in figure Figure 5. In this case, a record containing the first 64k (minus the length of the other fields) is exported using a template similar to the partial body template above. Then additional sections of the SIP body at subsequent specified offsets into the SIP body are exported in continuation records in subsequent messages. The SIP message whose body is continued is uniquely identified by the 5-tuple flow key and the observation time. This template allows the export of up to 65487 bytes of raw SIP message per IPFIX message (65515 message payload - 25 fixed fields - 3 sipMessageSection length).

```
observationTimeMilliseconds(323) [8]
sipMessageSectionOffset(35566/417) [4]
sourceIPv4Address(8) [4]
destinationIPv4Address(12) [4]
sourceTransportPort(7) [2]
destinationTransportPort(11) [2]
protocolIdentifier(4) [1]
sipMessageSection(35566/416) [v]
```

Figure 5: Template for Message Continuation

#### 4.2. Supporting fast searches of SIPCLF log files

The order of fields in an IPFIX data record is semantically unimportant, since the Template defines where each field in the record is. However, note that in all the Templates defined to this point, we have placed fixed-length fields before variable length fields. While logging processes are free to place the Information Elements in the Templates they export in any order, they SHOULD place fixed-length Information Elements before variable-length Information Elements. This increases the efficiency of both Message export and decoding, since each fixed length field as well as the first variable-length field have fixed offsets into the Message.

Random access to a field within a record (i.e., for comparison purposes during a sequential scan) is therefore constant-time for fixed-length fields, and requires one or two comparisons per subsequent field for variable-length fields.

However, since Sets encode the length of the Set, not of the records therein, and each Set may contain multiple records, a Set containing records with variable-length fields must be sequentially searched in order to find the beginning of the next Message. This situation MAY be improved by exporting only one Record per Set: in this case, the Set length equals the Record length minus the constant Set header length (4), reducing the number of comparisons needed for skipping records.

#### 5. Efficiency Analysis

[TODO]

#### 6. Examples

This section presents several views of an example IPFIX-based SIPCLF log, in order to increase understanding of what IPFIX would mean for SIPCLF. We present both binary and textual forms. The tools to generate this section are based upon the open-source ripfix [ripfix] implementation of IPFIX, maintained by one of the authors of this draft.

Here we show the log entries generated by the situations in sections 9.1 through 9.4 of [I-D.ietf-sipclf-problem-statement].

### 6.1. Base Template Export

A SIPCLF log file is, as described above just an IPFIX File with SIPCLF Templates. As Templates are themselves exported in IPFIX Messages, and a File is simply composed of a stream of Messages, first, we can export a Message containing only Templates describing the record formats we will use in subsequent examples. These Templates are derived from the base Templates as shown in Figure 2 and Figure 3, with the sipClientTransaction and sipServerTransaction Information Elements appended. We use two templates here, one each for request and response for IPv4

Exporting these Templates results in the following IPFIX message, illustrated as an annotated hexdump in Figure 6.

```

0000: 00 0a 00 fc 4c c0 2a a2 00 00 00 00 00 00 30 39  ....L.*.....09
      [ IPFIX message header, length 252 ]
0010: 00 02 00 ec                                     ....
      [ Template set (ID 2) header, length 236 ]
0014:          01 01 00 11 01 43 00 08 81 99 00 04          .....C.....
0020: 00 00 8a ee 00 08 00 04 00 0c 00 04 00 07 00 02  .....
0030: 00 0b 00 02 00 04 00 01 81 92 00 01 00 00 8a ee  .....
0040: 81 a3 00 01 00 00 8a ee 81 93 ff ff 00 00 8a ee  .....
0050: 81 96 ff ff 00 00 8a ee 81 97 ff ff 00 00 8a ee  .....
0060: 81 94 ff ff 00 00 8a ee 81 95 ff ff 00 00 8a ee  .....
0070: 81 98 ff ff 00 00 8a ee 81 9e ff ff 00 00 8a ee  .....
0080: 81 9d ff ff 00 00 8a ee                          .....
      [ Template 257, 17 elements (v4 request) ]
0088:          01 02 00 11 01 43 00 08          .....C..
0090: 81 99 00 04 00 00 8a ee 00 08 00 04 00 0c 00 04  .....
00a0: 00 07 00 02 00 0b 00 02 00 04 00 01 81 92 00 01  .....
00b0: 00 00 8a ee 81 a3 00 01 00 00 8a ee 81 9c 00 02  .....
00c0: 00 00 8a ee 81 96 ff ff 00 00 8a ee 81 97 ff ff  .....
00d0: 00 00 8a ee 81 94 ff ff 00 00 8a ee 81 95 ff ff  .....
00e0: 00 00 8a ee 81 98 ff ff 00 00 8a ee 81 9e ff ff  .....
00f0: 00 00 8a ee 81 9d ff ff 00 00 8a ee              .....
      [ Template 258, 17 elements (v4 response) ]

```

Figure 6: Base template message export

### 6.2. UAC registration

Having exported templates, now we create a simple SIPCLF-IPFIX log message representing a UAC registration as seen from the UAC, corresponding to example 9.1 in [I-D.ietf-sipclf-problem-statement]. This message contains two records, including the UAS registration request, and the response received. This is shown in the annotated hexdump in Figure 7.

```

0000: 00 0a 00 d8 4c 90 7f c1 00 00 00 00 00 30 39 ....L.....09
      [ IPFIX message header, length 218 ]
0010: 01 01 00 6b ...k
      [ Data set (ID 257) header, length 107 ]
0014: 00 00 01 29 13 66 13 93 00 00 00 01 ...).f.....
0020: c6 33 64 01 c6 33 64 0a 13 c4 13 c4 11 0c 02 0f .3d..3d.....
0030: 73 69 70 3a 65 78 61 6d 70 6c 65 2e 63 6f 6d 00 sip:example.com.
0040: 00 15 73 69 70 3a 61 6c 69 63 65 40 65 78 61 6d ..sip:alice@exam
0050: 70 6c 65 2e 63 6f 6d 05 37 36 79 68 68 15 66 38 ple.com.76yhh.f8
0060: 31 2d 64 34 2d 66 36 40 65 78 61 6d 70 6c 65 2e 1-d4-f6@example.
0070: 63 6f 6d 06 63 2d 74 72 2d 31 00 com.c-tr-1.
      [ Request record content ]
007b: 01 02 00 5d ...]
      [ Data set (ID 258) header, length 93 ]
007f: 00
0080: 00 01 29 13 66 15 24 00 00 00 01 c6 33 64 0a c6 ..).f.$.....3d..
0090: 33 64 01 13 c4 13 c4 11 0c 01 00 c8 00 00 15 73 3d.....s
00a0: 69 70 3a 61 6c 69 63 65 40 65 78 61 6d 70 6c 65 ip:alice@example
00b0: 2e 63 6f 6d 05 37 36 79 68 68 15 66 38 31 2d 64 .com.76yhh.f81-d
00c0: 34 2d 66 36 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 4-f6@example.com
00d0: 06 63 2d 74 72 2d 31 00 .c-tr-1.
      [ Response record content ]

```

Figure 7: Message containing two log entries for UAC registration

While this demonstrates the binary nature of the SIPCLF-IPFIX format, and shows the content framing for this message, it is not readable for illustration purposes. In Figure 8, as in all subsequent examples, we run the message through the rfdump tool provided with ripfix to provide a more human-readable view. Note that the sipMethod and sipObservationType are encoded according to the registries in Section 2.

```
==== message sequence 0 in domain 12345 at 2010-08-11 11:53:27 UTC ====
---- record 12345/257 ----
observationTimeMilliseconds => 2010-06-07 17:12:23 UTC
sipSequenceNumber => 1
sourceIPv4Address => 198.51.100.1
destinationIPv4Address => 198.51.100.10
sourceTransportPort => 5060
destinationTransportPort => 5060
protocolIdentifier => 17
sipMethod => 12
sipObservationType => 2
sipRequestURI => sip:example.com
sipToURI =>
sipToTag =>
sipFromURI => sip:alice@example.com
sipFromTag => 76yhh
sipCallId => f81-d4-f6@example.com
sipClientTransaction => c-tr-1
sipServerTransaction =>
---- record 12345/258 ----
observationTimeMilliseconds => 2010-06-07 17:12:24 UTC
sipSequenceNumber => 1
sourceIPv4Address => 198.51.100.10
destinationIPv4Address => 198.51.100.1
sourceTransportPort => 5060
destinationTransportPort => 5060
sipResponseStatus => 200
protocolIdentifier => 17
sipMethod => 12
sipObservationType => 1
sipToURI =>
sipToTag =>
sipFromURI => sip:alice@example.com
sipFromTag => 76yhh
sipCallId => f81-d4-f6@example.com
sipClientTransaction => c-tr-1
sipServerTransaction =>
```

Figure 8: Message containing two log entries for UAC registration

### 6.3. Direct Call

This example demonstrates the logging of a direct call from Alice to Bob, as seen by Bob's agent, corresponding to example 9.2 in [I-D.ietf-sipclf-problem-statement]. Here we have four records: an INVITE received from Alice, a 180 Ringing sent back followed by a 200 OK, and an ACK received from Alice. This is shown in the hexdump in Figure 9; message headers, set headers, and data records are



separated by '|' characters here for compactness. Note here that each record has its own data set as discussed in Section 4.2, even when two messages using the same are adjacent in the message.

```

0000: 00 0a 02 1a 4c c0 2c b3 00 00 00 00 00 00 30 39 ....L.,.....09
0010: 01 01 00 88 00 00 01 29 13 66 13 93 00 00 00 20 .....).f.....
0020: c6 33 64 01 cb 00 71 01 13 c4 13 c4 11 05 02 18 .3d...q.....
0030: 73 69 70 3a 62 6f 62 40 62 6f 62 31 2e 65 78 61 sip:bob@bob1.exa
0040: 6d 70 6c 65 2e 6e 65 74 13 73 69 70 3a 62 6f 62 mple.net.sip:bob
0050: 40 65 78 61 6d 70 6c 65 2e 6e 65 74 00 15 73 69 @example.net..si
0060: 70 3a 61 6c 69 63 65 40 65 78 61 6d 70 6c 65 2e p:alice@example.
0070: 63 6f 6d 05 37 36 79 68 68 15 66 38 32 2d 64 34 com.76yhh.f82-d4
0080: 2d 66 37 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 07 -f7@example.com.
0090: 63 2d 31 2d 78 74 36 00 01 02 00 79 00 00 01 29 c-1-xt6....y...)
00a0: 13 66 18 aa 00 00 00 20 cb 00 71 01 c6 33 64 01 .f..... .q..3d.
00b0: 13 c4 13 c4 11 05 01 00 b4 13 73 69 70 3a 62 6f .....sip:bo
00c0: 62 40 65 78 61 6d 70 6c 65 2e 6e 65 74 08 62 2d b@example.net.b-
00d0: 69 6e 36 2d 69 75 15 73 69 70 3a 61 6c 69 63 65 in6-iu.sip:alice
00e0: 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 05 37 36 79 @example.com.76y
00f0: 68 68 15 66 38 32 2d 64 34 2d 66 37 40 65 78 61 hh.f82-d4-f7@exa
0100: 6d 70 6c 65 2e 63 6f 6d 07 63 2d 31 2d 78 74 36 mple.com.c-1-xt6
0110: 00 01 02 00 79 00 00 01 29 13 66 1c f4 00 00 00 ....y...).f.....
0120: 20 cb 00 71 01 c6 33 64 01 13 c4 13 c4 11 05 01 ..q..3d.....
0130: 00 c8 13 73 69 70 3a 62 6f 62 40 65 78 61 6d 70 ...sip:bob@examp
0140: 6c 65 2e 6e 65 74 08 62 2d 69 6e 36 2d 69 75 15 le.net.b-in6-iu.
0150: 73 69 70 3a 61 6c 69 63 65 40 65 78 61 6d 70 6c sip:alice@examp
0160: 65 2e 63 6f 6d 05 37 36 79 68 68 15 66 38 32 2d e.com.76yhh.f82-
0170: 64 34 2d 66 37 40 65 78 61 6d 70 6c 65 2e 63 6f d4-f7@example.co
0180: 6d 07 63 2d 31 2d 78 74 36 00 01 01 00 90 00 00 m.c-1-xt6.....
0190: 01 29 13 66 1d 08 00 00 00 20 c6 33 64 01 cb 00 .).f..... .3d...
01a0: 71 01 13 c4 13 c4 11 01 02 18 73 69 70 3a 62 6f q.....sip:bo
01b0: 62 40 62 6f 62 31 2e 65 78 61 6d 70 6c 65 2e 6e b@bob1.example.n
01c0: 65 74 13 73 69 70 3a 62 6f 62 40 65 78 61 6d 70 et.sip:bob@examp
01d0: 6c 65 2e 6e 65 74 08 62 2d 69 6e 36 2d 69 75 15 le.net.b-in6-iu.
01e0: 73 69 70 3a 61 6c 69 63 65 40 65 78 61 6d 70 6c sip:alice@examp
01f0: 65 2e 63 6f 6d 05 37 36 79 68 68 15 66 38 32 2d e.com.76yhh.f82-
0200: 64 34 2d 66 37 40 65 78 61 6d 70 6c 65 2e 63 6f d4-f7@example.co
0210: 6d 07 63 2d 31 2d 78 74 36 00 m.c-1-xt6.

```

Figure 9: Message containing four log entries for a simple call

#### 6.4. Single Downstream Branch Call

The example in Figure 10 demonstrates the logging of a call with a downstream branch to Bob, as seen by the proxy which the call traverses, corresponding to example 9.3 in [I-D.ietf-sipclf-problem-statement]. See this example in the problem statement for more details.

```
0000: 00 0a 04 e1 4c c0 2c e5 00 00 00 00 00 00 30 39 ....L.,.....09
0010: 01 01 00 7e 00 00 01 29 13 66 13 93 00 00 00 2b ...~...).f.....+
0020: c6 33 64 01 c6 33 64 0a 13 c4 13 c4 11 05 01 13 .3d..3d.....
0030: 73 69 70 3a 62 6f 62 40 65 78 61 6d 70 6c 65 2e sip:bob@example.
0040: 6e 65 74 13 73 69 70 3a 62 6f 62 40 65 78 61 6d net.sip:bob@exam
0050: 70 6c 65 2e 6e 65 74 00 15 73 69 70 3a 61 6c 69 ple.net..sip:ali
0060: 63 65 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 04 61 ce@example.com.a
0070: 6c 2d 31 12 74 72 2d 38 37 68 40 65 78 61 6d 70 l-1.tr-87h@examp
0080: 6c 65 2e 63 6f 6d 00 06 73 2d 78 2d 74 72 01 02 le.com..s-x-tr..
0090: 00 6c 00 00 01 29 13 66 14 c1 00 00 00 2b c6 33 .l...).f.....+3
00a0: 64 0a c6 33 64 01 13 c4 13 c4 11 05 02 00 64 13 d..3d.....d.
00b0: 73 69 70 3a 62 6f 62 40 65 78 61 6d 70 6c 65 2e sip:bob@example.
00c0: 6e 65 74 00 15 73 69 70 3a 61 6c 69 63 65 40 65 net..sip:alice@e
00d0: 78 61 6d 70 6c 65 2e 63 6f 6d 04 61 6c 2d 31 12 xample.com.al-1.
00e0: 74 72 2d 38 37 68 40 65 78 61 6d 70 6c 65 2e 63 tr-87h@example.c
00f0: 6f 6d 00 06 73 2d 78 2d 74 72 01 01 00 89 00 00 om..s-x-tr.....
0100: 01 29 13 66 18 a6 00 00 00 2b c6 33 64 0a cb 00 .).f.....+3d...
0110: 71 01 13 c4 13 c4 11 05 02 18 73 69 70 3a 62 6f q.....sip:bo
0120: 62 40 62 6f 62 31 2e 65 78 61 6d 70 6c 65 2e 6e b@bob1.example.n
0130: 65 74 13 73 69 70 3a 62 6f 62 40 65 78 61 6d 70 et.sip:bob@examp
0140: 6c 65 2e 6e 65 74 00 15 73 69 70 3a 61 6c 69 63 le.net..sip:alic
0150: 65 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 04 61 6c e@example.com.al
0160: 2d 31 12 74 72 2d 38 37 68 40 65 78 61 6d 70 6c -1.tr-87h@exampl
0170: 65 2e 63 6f 6d 06 63 2d 78 2d 74 72 06 73 2d 78 e.com.c-x-tr.s-x
0180: 2d 74 72 01 02 00 76 00 00 01 29 13 66 19 70 00 -tr...v...).f.p.
0190: 00 00 2b cb 00 71 01 c6 33 64 0a 13 c4 13 c4 11 ..+..q..3d.....
01a0: 05 01 00 64 13 73 69 70 3a 62 6f 62 40 65 78 61 ...d.sip:bob@exa
01b0: 6d 70 6c 65 2e 6e 65 74 04 62 31 2d 31 15 73 69 mple.net.b1-1.si
01c0: 70 3a 61 6c 69 63 65 40 65 78 61 6d 70 6c 65 2e p:alice@example.
01d0: 63 6f 6d 04 61 6c 2d 31 12 74 72 2d 38 37 68 40 com.al-1.tr-87h@
01e0: 65 78 61 6d 70 6c 65 2e 63 6f 6d 06 63 2d 78 2d example.com.c-x-
01f0: 74 72 06 73 2d 78 2d 74 72 01 02 00 76 00 00 01 tr.s-x-tr...v...
0200: 29 13 66 1b c8 00 00 00 2b cb 00 71 01 c6 33 64 ).f.....+..q..3d
0210: 0a 13 c4 13 c4 11 05 01 00 b4 13 73 69 70 3a 62 .....sip:b
0220: 6f 62 40 65 78 61 6d 70 6c 65 2e 6e 65 74 04 62 ob@example.net.b
0230: 31 2d 31 15 73 69 70 3a 61 6c 69 63 65 40 65 78 1-1.sip:alice@ex
0240: 61 6d 70 6c 65 2e 63 6f 6d 04 61 6c 2d 31 12 74 ample.com.al-1.t
0250: 72 2d 38 37 68 40 65 78 61 6d 70 6c 65 2e 63 6f r-87h@example.co
0260: 6d 06 63 2d 78 2d 74 72 06 73 2d 78 2d 74 72 01 m.c-x-tr.s-x-tr.
0270: 02 00 76 00 00 01 29 13 66 1c 98 00 00 00 2b c6 ..v...).f.....+.
0280: 33 64 0a c6 33 64 01 13 c4 13 c4 11 05 02 00 b4 3d..3d.....
0290: 13 73 69 70 3a 62 6f 62 40 65 78 61 6d 70 6c 65 .sip:bob@example
02a0: 2e 6e 65 74 04 62 31 2d 31 15 73 69 70 3a 61 6c .net.b1-1.sip:al
02b0: 69 63 65 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 04 ice@example.com.
02c0: 61 6c 2d 31 12 74 72 2d 38 37 68 40 65 78 61 6d al-1.tr-87h@exam
02d0: 70 6c 65 2e 63 6f 6d 06 63 2d 78 2d 74 72 06 73 ple.com.c-x-tr.s
02e0: 2d 78 2d 74 72 01 02 00 76 00 00 01 29 13 66 20 -x-tr...v...).f
02f0: f0 00 00 00 2b cb 00 71 01 c6 33 64 0a 13 c4 13 ....+..q..3d....
```

```

0300: c4 11 05 01 00 c8 13 73 69 70 3a 62 6f 62 40 65 .....sip:bob@e
0310: 78 61 6d 70 6c 65 2e 6e 65 74 04 62 31 2d 31 15 xample.net.b1-1.
0320: 73 69 70 3a 61 6c 69 63 65 40 65 78 61 6d 70 6c sip:alice@exampl
0330: 65 2e 63 6f 6d 04 61 6c 2d 31 12 74 72 2d 38 37 e.com.al-1.tr-87
0340: 68 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 06 63 2d h@example.com.c-
0350: 78 2d 74 72 06 73 2d 78 2d 74 72 01 02 00 76 00 x-tr.s-x-tr...v.
0360: 00 01 29 13 66 21 a4 00 00 00 2b c6 33 64 0a c6 ..).f!.....+3d..
0370: 33 64 01 13 c4 13 c4 11 05 02 00 c8 13 73 69 70 3d.....sip
0380: 3a 62 6f 62 40 65 78 61 6d 70 6c 65 2e 6e 65 74 :bob@example.net
0390: 04 62 31 2d 31 15 73 69 70 3a 61 6c 69 63 65 40 .b1-1.sip:alice@
03a0: 65 78 61 6d 70 6c 65 2e 63 6f 6d 04 61 6c 2d 31 example.com.al-1
03b0: 12 74 72 2d 38 37 68 40 65 78 61 6d 70 6c 65 2e .tr-87h@example.
03c0: 63 6f 6d 06 63 2d 78 2d 74 72 06 73 2d 78 2d 74 com.c-x-tr.s-x-t
03d0: 72 01 01 00 88 00 00 01 29 13 66 28 ac 00 00 00 r.....).f(....
03e0: 2b c6 33 64 01 c6 33 64 0a 13 c4 13 c4 11 01 01 +.3d..3d.....
03f0: 13 73 69 70 3a 62 6f 62 40 65 78 61 6d 70 6c 65 .sip:bob@example
0400: 2e 6e 65 74 13 73 69 70 3a 62 6f 62 40 65 78 61 .net.sip:bob@exa
0410: 6d 70 6c 65 2e 6e 65 74 04 62 31 2d 31 15 73 69 mple.net.b1-1.si
0420: 70 3a 61 6c 69 63 65 40 65 78 61 6d 70 6c 65 2e p:alice@example.
0430: 63 6f 6d 04 61 6c 2d 31 12 74 72 2d 38 37 68 40 com.al-1.tr-87h@
0440: 65 78 61 6d 70 6c 65 2e 63 6f 6d 06 63 2d 78 2d example.com.c-x-
0450: 74 72 06 73 2d 78 2d 74 72 01 01 00 88 00 00 01 tr.s-x-tr.....
0460: 29 13 66 28 ac 00 00 00 2b c6 33 64 0a cb 00 71 ).f(.....+3d...q
0470: 01 13 c4 13 c4 11 01 02 13 73 69 70 3a 62 6f 62 .....sip:bob
0480: 40 65 78 61 6d 70 6c 65 2e 6e 65 74 13 73 69 70 @example.net.sip
0490: 3a 62 6f 62 40 65 78 61 6d 70 6c 65 2e 6e 65 74 :bob@example.net
04a0: 04 62 31 2d 31 15 73 69 70 3a 61 6c 69 63 65 40 .b1-1.sip:alice@
04b0: 65 78 61 6d 70 6c 65 2e 63 6f 6d 04 61 6c 2d 31 example.com.al-1
04c0: 12 74 72 2d 38 37 68 40 65 78 61 6d 70 6c 65 2e .tr-87h@example.
04d0: 63 6f 6d 06 63 2d 78 2d 74 72 06 73 2d 78 2d 74 com.c-x-tr.s-x-t
04e0: 72

```

Figure 10: Message containing ten log entries for a downstream branch call

## 6.5. Forked Call

The example in Figure 12 demonstrates the logging of forked call to Bob, as seen by one of Bob's instances which forks the call traverses, corresponding to example 9.4 in [I-D.ietf-sipclf-problem-statement]. See this example for more details. Note that, since Bob's first instance is multihomed IPv4-IPv6, this example requires additional templates: request and response templates for IPv4 to IPv6 and back, these are shown in Figure 11.

```

0000: 00 0a 01 e4 4c c0 2d 9b 00 00 00 00 00 30 39 ....L.-.....09
0010: 00 02 01 d4 01 05 00 11 01 43 00 08 81 99 00 04 .....C.....
0020: 00 00 8a ee 00 08 00 04 00 1c 00 10 00 07 00 02 .....
0030: 00 0b 00 02 00 04 00 01 81 92 00 01 00 00 8a ee .....
0040: 81 a3 00 01 00 00 8a ee 81 93 ff ff 00 00 8a ee .....
0050: 81 96 ff ff 00 00 8a ee 81 97 ff ff 00 00 8a ee .....
0060: 81 94 ff ff 00 00 8a ee 81 95 ff ff 00 00 8a ee .....
0070: 81 98 ff ff 00 00 8a ee 81 9e ff ff 00 00 8a ee .....
0080: 81 9d ff ff 00 00 8a ee 01 06 00 11 01 43 00 08 .....C..
0090: 81 99 00 04 00 00 8a ee 00 08 00 04 00 1c 00 10 .....
00a0: 00 07 00 02 00 0b 00 02 00 04 00 01 81 92 00 01 .....
00b0: 00 00 8a ee 81 a3 00 01 00 00 8a ee 81 9c 00 02 .....
00c0: 00 00 8a ee 81 96 ff ff 00 00 8a ee 81 97 ff ff .....
00d0: 00 00 8a ee 81 94 ff ff 00 00 8a ee 81 95 ff ff .....
00e0: 00 00 8a ee 81 98 ff ff 00 00 8a ee 81 9e ff ff .....
00f0: 00 00 8a ee 81 9d ff ff 00 00 8a ee 01 07 00 11 .....
0100: 01 43 00 08 81 99 00 04 00 00 8a ee 00 1b 00 10 .C.....
0110: 00 0c 00 04 00 07 00 02 00 0b 00 02 00 04 00 01 .....
0120: 81 92 00 01 00 00 8a ee 81 a3 00 01 00 00 8a ee .....
0130: 81 93 ff ff 00 00 8a ee 81 96 ff ff 00 00 8a ee .....
0140: 81 97 ff ff 00 00 8a ee 81 94 ff ff 00 00 8a ee .....
0150: 81 95 ff ff 00 00 8a ee 81 98 ff ff 00 00 8a ee .....
0160: 81 9e ff ff 00 00 8a ee 81 9d ff ff 00 00 8a ee .....
0170: 01 08 00 11 01 43 00 08 81 99 00 04 00 00 8a ee .....C.....
0180: 00 1b 00 10 00 0c 00 04 00 07 00 02 00 0b 00 02 .....
0190: 00 04 00 01 81 92 00 01 00 00 8a ee 81 a3 00 01 .....
01a0: 00 00 8a ee 81 9c 00 02 00 00 8a ee 81 96 ff ff .....
01b0: 00 00 8a ee 81 97 ff ff 00 00 8a ee 81 94 ff ff .....
01c0: 00 00 8a ee 81 95 ff ff 00 00 8a ee 81 98 ff ff .....
01d0: 00 00 8a ee 81 9e ff ff 00 00 8a ee 81 9d ff ff .....
01e0: 00 00 8a ee .....

```

Figure 11: Message containing templates for IPv4 to IPv6 requests and responses, and vice versa

```

0000: 00 0a 07 8c 4c c0 2d 9b 00 00 00 00 00 30 39 ....L.-.....09
0010: 01 01 00 7e 00 00 01 29 13 66 13 93 00 00 00 2b ...~...)..f.....+
0020: c6 33 64 01 cb 00 71 c8 13 c4 13 c4 11 05 01 13 .3d...q.....
0030: 73 69 70 3a 62 6f 62 40 65 78 61 6d 70 6c 65 2e sip:bob@example.
0040: 6e 65 74 13 73 69 70 3a 62 6f 62 40 65 78 61 6d net.sip:bob@exam
0050: 70 6c 65 2e 6e 65 74 00 15 73 69 70 3a 61 6c 69 ple.net..sip:ali
0060: 63 65 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 04 61 ce@example.com.a
0070: 31 2d 31 12 74 72 2d 38 38 68 40 65 78 61 6d 70 1-1.tr-88h@examp
0080: 6c 65 2e 63 6f 6d 00 06 73 2d 31 2d 74 72 01 02 le.com..s-1-tr..
0090: 00 6c 00 00 01 29 13 66 14 c1 00 00 00 2b cb 00 .l...)..f.....+..
00a0: 71 c8 c6 33 64 01 13 c4 13 c4 11 05 02 00 64 13 q..3d.....d.
00b0: 73 69 70 3a 62 6f 62 40 65 78 61 6d 70 6c 65 2e sip:bob@example.

```

```
00c0: 6e 65 74 00 15 73 69 70 3a 61 6c 69 63 65 40 65 net..sip:alice@e
00d0: 78 61 6d 70 6c 65 2e 63 6f 6d 04 61 31 2d 31 12 xample.com.a1-1.
00e0: 74 72 2d 38 38 68 40 65 78 61 6d 70 6c 65 2e 63 tr-88h@example.c
00f0: 6f 6d 00 06 73 2d 31 2d 74 72 01 01 00 89 00 00 om..s-1-tr.....
0100: 01 29 13 66 18 a6 00 00 00 2b cb 00 71 c8 cb 00 .).f.....+.q...
0110: 71 01 13 c4 13 c4 11 05 02 18 73 69 70 3a 62 6f q.....sip:bo
0120: 62 40 62 6f 62 31 2e 65 78 61 6d 70 6c 65 2e 6e b@bob1.example.n
0130: 65 74 13 73 69 70 3a 62 6f 62 40 65 78 61 6d 70 et.sip:bob@examp
0140: 6c 65 2e 6e 65 74 00 15 73 69 70 3a 61 6c 69 63 le.net..sip:alic
0150: 65 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 04 61 31 e@example.com.a1
0160: 2d 31 12 74 72 2d 38 38 68 40 65 78 61 6d 70 6c -1.tr-88h@exampl
0170: 65 2e 63 6f 6d 06 63 2d 31 2d 74 72 06 73 2d 31 e.com.c-1-tr.s-1
0180: 2d 74 72 01 05 00 95 00 00 01 29 13 66 1a 9c 00 -tr.....).f...
0190: 00 00 2b cb 00 71 c8 20 01 0d b8 00 00 00 00 00 ..+.q. ....
01a0: 00 00 00 00 00 00 09 13 c4 13 c4 11 05 02 18 73 .....s
01b0: 69 70 3a 62 6f 62 40 62 6f 62 32 2e 65 78 61 6d ip:bob@bob2.exam
01c0: 70 6c 65 2e 6e 65 74 13 73 69 70 3a 62 6f 62 40 ple.net.sip:bob@
01d0: 65 78 61 6d 70 6c 65 2e 6e 65 74 00 15 73 69 70 example.net..sip
01e0: 3a 61 6c 69 63 65 40 65 78 61 6d 70 6c 65 2e 63 :alice@example.c
01f0: 6f 6d 04 61 31 2d 31 12 74 72 2d 38 38 68 40 65 om.a1-1.tr-88h@e
0200: 78 61 6d 70 6c 65 2e 63 6f 6d 06 63 2d 32 2d 74 xample.com.c-2-t
0210: 72 06 73 2d 31 2d 74 72 01 02 00 76 00 00 01 29 r.s-1-tr...v...)
0220: 13 66 1b c8 00 00 0b cb 00 71 01 cb 00 71 c8 .f.....+.q...q.
0230: 13 c4 13 c4 11 05 01 00 64 13 73 69 70 3a 62 6f .....d.sip:bo
0240: 62 40 65 78 61 6d 70 6c 65 2e 6e 65 74 04 62 31 b@example.net.b1
0250: 2d 31 15 73 69 70 3a 61 6c 69 63 65 40 65 78 61 -1.sip:alice@exa
0260: 6d 70 6c 65 2e 63 6f 6d 04 61 31 2d 31 12 74 72 mple.com.a1-1.tr
0270: 2d 38 38 68 40 65 78 61 6d 70 6c 65 2e 63 6f 6d -88h@example.com
0280: 06 63 2d 31 2d 74 72 06 73 2d 31 2d 74 72 01 08 .c-1-tr.s-1-tr..
0290: 00 82 00 00 01 29 13 66 1c f4 00 00 00 2b 20 01 .....).f.....+ .
02a0: 0d b8 00 00 00 00 00 00 00 00 00 00 09 cb 00 .....
02b0: 71 c8 13 c4 13 c4 11 05 01 00 64 13 73 69 70 3a q.....d.sip:
02c0: 62 6f 62 40 65 78 61 6d 70 6c 65 2e 6e 65 74 04 bob@example.net.
02d0: 62 32 2d 32 15 73 69 70 3a 61 6c 69 63 65 40 65 b2-2.sip:alice@e
02e0: 78 61 6d 70 6c 65 2e 63 6f 6d 04 61 31 2d 31 12 xample.com.a1-1.
02f0: 74 72 2d 38 38 68 40 65 78 61 6d 70 6c 65 2e 63 tr-88h@example.c
0300: 6f 6d 06 63 2d 32 2d 74 72 06 73 2d 31 2d 74 72 om.c-2-tr.s-1-tr
0310: 01 08 00 82 00 00 01 29 13 66 1f 4c 00 00 00 2b .....).f.L...+
0320: 20 01 0d b8 00 00 00 00 00 00 00 00 00 00 09 .....
0330: cb 00 71 c8 13 c4 13 c4 11 05 01 00 b4 13 73 69 ..q.....si
0340: 70 3a 62 6f 62 40 65 78 61 6d 70 6c 65 2e 6e 65 p:bob@example.ne
0350: 74 04 62 32 2d 32 15 73 69 70 3a 61 6c 69 63 65 t.b2-2.sip:alice
0360: 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 04 61 31 2d @example.com.a1-
0370: 31 12 74 72 2d 38 38 68 40 65 78 61 6d 70 6c 65 1.tr-88h@example
0380: 2e 63 6f 6d 06 63 2d 32 2d 74 72 06 73 2d 31 2d .com.c-2-tr.s-1-
0390: 74 72 01 02 00 72 00 00 01 29 13 66 20 6e 00 00 tr...r...).f n..
03a0: 00 2b cb 00 71 c8 c6 33 64 01 13 c4 13 c4 11 05 .+.q..3d.....
03b0: 02 00 b4 13 73 69 70 3a 62 6f 62 40 65 78 61 6d ....sip:bob@exam
```

```
03c0: 70 6c 65 2e 6e 65 74 00 15 73 69 70 3a 61 6c 69 ple.net..sip:ali
03d0: 63 65 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 04 61 ce@example.com.a
03e0: 31 2d 31 12 74 72 2d 38 38 68 40 65 78 61 6d 70 1-1.tr-88h@examp
03f0: 6c 65 2e 63 6f 6d 06 63 2d 32 2d 74 72 06 73 2d le.com.c-2-tr.s-
0400: 31 2d 74 72 01 02 00 76 00 00 01 29 13 66 21 a4 1-tr...v...).f!.
0410: 00 00 00 2b cb 00 71 c8 c6 33 64 01 13 c4 13 c4 ...+...q..3d.....
0420: 11 05 02 00 b4 13 73 69 70 3a 62 6f 62 40 65 78 .....sip:bob@ex
0430: 61 6d 70 6c 65 2e 6e 65 74 04 62 31 2d 31 15 73 ample.net.b1-1.s
0440: 69 70 3a 61 6c 69 63 65 40 65 78 61 6d 70 6c 65 ip:alice@example
0450: 2e 63 6f 6d 04 61 31 2d 31 12 74 72 2d 38 38 68 .com.a1-1.tr-88h
0460: 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 06 63 2d 31 @example.com.c-1
0470: 2d 74 72 06 73 2d 31 2d 74 72 01 02 00 76 00 00 -tr.s-1-tr...v...
0480: 01 29 13 66 23 98 00 00 00 2b cb 00 71 01 cb 00 .).f#....+...q...
0490: 71 c8 13 c4 13 c4 11 05 01 00 c8 13 73 69 70 3a q.....sip:
04a0: 62 6f 62 40 65 78 61 6d 70 6c 65 2e 6e 65 74 04 bob@example.net.
04b0: 62 31 2d 31 15 73 69 70 3a 61 6c 69 63 65 40 65 b1-1.sip:alice@e
04c0: 78 61 6d 70 6c 65 2e 63 6f 6d 04 61 31 2d 31 12 xample.com.a1-1.
04d0: 74 72 2d 38 38 68 40 65 78 61 6d 70 6c 65 2e 63 tr-88h@example.c
04e0: 6f 6d 06 63 2d 31 2d 74 72 06 73 2d 31 2d 74 72 om.c-1-tr.s-1-tr
04f0: 01 02 00 76 00 00 01 29 13 66 24 60 00 00 00 2b ...v...).f$'...+
0500: cb 00 71 c8 c6 33 64 01 13 c4 13 c4 11 05 02 00 ..q..3d.....
0510: c8 13 73 69 70 3a 62 6f 62 40 65 78 61 6d 70 6c ..sip:bob@exempl
0520: 65 2e 6e 65 74 04 62 31 2d 31 15 73 69 70 3a 61 e.net.b1-1.sip:a
0530: 6c 69 63 65 40 65 78 61 6d 70 6c 65 2e 63 6f 6d lice@example.com
0540: 04 61 31 2d 31 12 74 72 2d 38 38 68 40 65 78 61 .a1-1.tr-88h@exa
0550: 6d 70 6c 65 2e 63 6f 6d 06 63 2d 31 2d 74 72 06 mple.com.c-1-tr.
0560: 73 2d 31 2d 74 72 01 05 00 95 00 00 01 29 13 66 s-1-tr.....).f
0570: 25 29 00 00 00 2b cb 00 71 c8 20 01 0d b8 00 00 %)....+...q. ....
0580: 00 00 00 00 00 00 00 00 00 09 13 c4 13 c4 11 03 .....
0590: 02 18 73 69 70 3a 62 6f 62 40 62 6f 62 32 2e 65 ..sip:bob@bob2.e
05a0: 78 61 6d 70 6c 65 2e 6e 65 74 13 73 69 70 3a 62 xample.net.sip:b
05b0: 6f 62 40 65 78 61 6d 70 6c 65 2e 6e 65 74 00 15 ob@example.net..
05c0: 73 69 70 3a 61 6c 69 63 65 40 65 78 61 6d 70 6c sip:alice@exempl
05d0: 65 2e 63 6f 6d 04 61 31 2d 31 12 74 72 2d 38 38 e.com.a1-1.tr-88
05e0: 68 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 06 63 2d h@example.com.c-
05f0: 32 2d 74 72 06 73 2d 31 2d 74 72 01 08 00 7e 00 2-tr.s-1-tr...~.
0600: 00 01 29 13 66 28 3f 00 00 00 2b 20 01 0d b8 00 ..).f(?...+ ....
0610: 00 00 00 00 00 00 00 00 00 09 cb 00 71 c8 13 .....q..
0620: c4 13 c4 11 05 01 01 e7 13 73 69 70 3a 62 6f 62 .....sip:bob
0630: 40 65 78 61 6d 70 6c 65 2e 6e 65 74 00 15 73 69 @example.net..si
0640: 70 3a 61 6c 69 63 65 40 65 78 61 6d 70 6c 65 2e p:alice@example.
0650: 63 6f 6d 04 61 31 2d 31 12 74 72 2d 38 38 68 40 com.a1-1.tr-88h@
0660: 65 78 61 6d 70 6c 65 2e 63 6f 6d 06 63 2d 32 2d example.com.c-2-
0670: 74 72 06 73 2d 31 2d 74 72 01 05 00 95 00 00 01 tr.s-1-tr.....
0680: 29 13 66 2a 0f 00 00 00 2b cb 00 71 c8 20 01 0d ).f*....+...q. ..
0690: b8 00 00 00 00 00 00 00 00 09 13 c4 13 .....
06a0: c4 11 01 02 18 73 69 70 3a 62 6f 62 40 62 6f 62 .....sip:bob@bob
06b0: 32 2e 65 78 61 6d 70 6c 65 2e 6e 65 74 13 73 69 2.example.net.si
```

```

06c0: 70 3a 62 6f 62 40 65 78 61 6d 70 6c 65 2e 6e 65 p:bob@example.ne
06d0: 74 00 15 73 69 70 3a 61 6c 69 63 65 40 65 78 61 t..sip:alice@exa
06e0: 6d 70 6c 65 2e 63 6f 6d 04 61 31 2d 31 12 74 72 mple.com.a1-1.tr
06f0: 2d 38 38 68 40 65 78 61 6d 70 6c 65 2e 63 6f 6d -88h@example.com
0700: 06 63 2d 32 2d 74 72 06 73 2d 31 2d 74 72 01 08 .c-2-tr.s-1-tr..
0710: 00 7e 00 00 01 29 13 66 2c 31 00 00 00 2b 20 01 .~...).f,1...+ .
0720: 0d b8 00 00 00 00 00 00 00 00 00 00 00 09 cb 00 .....
0730: 71 c8 13 c4 13 c4 11 03 01 00 c8 13 73 69 70 3a q.....sip:
0740: 62 6f 62 40 65 78 61 6d 70 6c 65 2e 6e 65 74 00 bob@example.net.
0750: 15 73 69 70 3a 61 6c 69 63 65 40 65 78 61 6d 70 .sip:alice@examp
0760: 6c 65 2e 63 6f 6d 04 61 31 2d 31 12 74 72 2d 38 le.com.a1-1.tr-8
0770: 38 68 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 06 63 8h@example.com.c
0780: 2d 32 2d 74 72 06 73 2d 31 2d 74 72 -2-tr.s-1-tr

```

Figure 12: Message containing sixteen log entries for a forked call

## 6.6. Torture Tests

[EDITOR'S NOTE: The torture tests have to be reworked to reflect proper handling of escapes: we log things escaped, and require the log reader to follow the SIP escaping rules. However, note that this makes the torture tests much less relevant with respect to the logging format, and we should review whether it makes sense to retain this section.]

To demonstrate that the SIPCLF-IPFIX file format can handle other than simple situations, here we show how log entries from selected SIP torture tests given in [RFC4475]. Note that this demonstrates only that there is nothing inherent to the logging format that precludes handling these cases; implementors of logging using SIPCLF-IPFIX must still take care that the SIP-specific side of the implementation handles these torture tests correctly. Each of these tests displays a resulting record with templates defined in Section 6.1 and values for unspecified fields taken from Section 6.2.

The torture test in section 3.1.1.2 tests the handling of unusual characters. Assuming that the SIP parser is able to handle this message, the resulting log entry in rfdump format is shown in Figure 13 (hand-edited to fit long lines within Internet-Draft limits). Note that the unknown SIP method is logged as such (method 0 = Unknown).

```
==== message sequence 0 in domain 12345 at 2010-08-12 07:44:33 UTC ====
---- record 12345/257 ----
observationTimeMilliseconds => 2010-08-12 07:44:33 UTC
sipSequenceNumber => 139122385
sourceIPv4Address => 198.51.100.10
destinationIPv4Address => 198.51.100.1
sourceTransportPort => 5060
destinationTransportPort => 5060
protocolIdentifier => 17
sipMethod => 0
sipObservationType => 1
sipRequestURI => sip:1_unusual.URI~(to-be!sure)&isn't+it$/crazy?,/
;;*:&it+has=1,weird!*pas$wo~d_too.(doesn't-it)@example.com
sipToURI => sip:1_unusual.URI~(to-be!sure)&isn't+it$/crazy?,/
;;*@example.com
sipToTag =>
sipFromURI => sip:mundane@example.com
sipFromTag => _token~1'+`*%!-.
sipCallId => intmeth.word%ZK-!.*_+'@word`)(><:\/"[?]{
sipClientTransaction =>
sipServerTransaction =>
```

Figure 13: Message containing wide variety of characters

The torture test in section 3.1.1.4 tests the handling of embedded NULs in strings. Since IPFIX uses length-prefixing for variable length strings, this does not present a problem for logging. The resulting record rfdump format in Figure 14. Here, hex escaping shows that the NUL is properly embedded in the string.



```

==== message sequence 0 in domain 12345 at 2010-08-11 13:01:05 UTC ====
---- record 12345/257 ----
observationTimeMilliseconds => 2010-08-11 13:01:05 UTC
sipSequenceNumber => 14398234
sourceIPv4Address => 198.51.100.10
destinationIPv4Address => 198.51.100.1
sourceTransportPort => 5060
destinationTransportPort => 5060
protocolIdentifier => 17
sipMethod => 12
sipObservationType => 2
sipRequestURI => sip:example.com
sipToURI => sip:null-\x00-null@example.com
sipToTag =>
sipFromURI => sip:null-\x00-null@example.com
sipFromTag => 839923423
sipCallId => escnull.39203ndfvkjdasfkq3w4otr0adsfdfnvd
sipClientTransaction =>
sipServerTransaction =>

```

Figure 14: Message containing NULs in URIs

The sipToURI portion of the record is shown in Figure 15 as an annotated hexdump. Note the length byte at the beginning of the string, and the NUL embedded inside.

```

003f:                                     1b      .
                                [ varlen record, length 27 ]
0040: 73 69 70 3a 6e 75 6c 6c 2d 00 2d 6e 75 6c 6c 40  sip:null-.-null@
                                [ note NUL here ^^ ]
0050: 65 78 61 6d 70 6c 65 2e 63 6f 6d                example.com

```

Figure 15: Message containing NULs in URIs

## 7. Security Considerations

[TODO]

## 8. IANA Considerations

[TODO: add new SIP IEs to, create new SIP Method registry, or add numbering to existing registry.]

## 9. Acknowledgments

Thanks to Cullen Jennings for his provided insightful discussions, specific comments and much needed corrections, and to Nico d'Heureuse for his help with the RFC 3665 examples.

## 10. References

### 10.1. Normative References

- [RFC5101] Claise, B., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.
- [RFC5655] Trammell, B., Boschi, E., Mark, L., Zseby, T., and A. Wagner, "Specification of the IP Flow Information Export (IPFIX) File Format", RFC 5655, October 2009.

### 10.2. Informative 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)",  
draft-ietf-sipclf-problem-statement-03 (work in progress),  
June 2010.
- [I-D.kaplan-dispatch-session-id]  
Kaplan, H., "A Session Identifier for the Session Initiation Protocol (SIP)",  
draft-kaplan-dispatch-session-id-02 (work in progress),  
July 2010.
- [I-D.trammell-ipfix-text-iespec]  
Trammell, B., "A Lightweight Textual Format for IPFIX Information Models and Templates",  
draft-trammell-ipfix-text-iespec-01 (work in progress),  
August 2010.
- [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.
- [RFC3665] Johnston, A., Donovan, S., Sparks, R., Cunningham, C., and K. Summers, "Session Initiation Protocol (SIP) Basic Call Flow Examples", BCP 75, RFC 3665, December 2003.

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

[ripfix] Trammell, B., "ripfix: IPFIX for Ruby", available at <http://ripfix.rubyforge.org/>.

#### Appendix A. Example messages in base64

This section contains the example messages from this revision of this draft in base64 encoding, for ease of processing by automated tools.

The base templates are in this message:

```
AAoA/EzALZsAAAAAAAwOQACAOWBAQARAUMACIGZAAQAAIruAAgABAAMAAQA
BwACAAAsAAgAEAAGBkgABAACK7oGjAAEAAIrugZP//wAAiu6Blv//AACK7oGX
//8AAIrugZT//wAAiu6Blf//AACK7oGY//8AAIrugZ7//wAAiu6Bnf//AACK
7gECABEBQwAIgZkABAAAiu4ACAAEAwABAHAHAACwACAAQAAYGSAAEAAIru
gaMAAQAAiu6BnAACAAACK7oGW//8AAIrugZf//wAAiu6BlP//AACK7oGV//8A
AIrugZj//wAAiu6Bnv//AACK7oGd//8AAIru
```

The extended 4to6 and 6to4 templates are in this message:

```
AAoB5EzALZsAAAAAAAwOQACAdQBBQARAUMACIGZAAQAAIruAAgABAACABAA
BwACAAAsAAgAEAAGBkgABAACK7oGjAAEAAIrugZP//wAAiu6Blv//AACK7oGX
//8AAIrugZT//wAAiu6Blf//AACK7oGY//8AAIrugZ7//wAAiu6Bnf//AACK
7gEGABEBQwAIgZkABAAAiu4ACAAEAwABAHAHAACwACAAQAAYGSAAEAAIru
gaMAAQAAiu6BnAACAAACK7oGW//8AAIrugZf//wAAiu6BlP//AACK7oGV//8A
AIrugZj//wAAiu6Bnv//AACK7oGd//8AAIruAQcAEQFDAAiBmQAEAAACK7gAb
ABAADAAEAACAAgALAAIABAABgZIAAQAAiu6BowABAACK7oGT//8AAIrugZb/
/wAAiu6Bl///AACK7oGU//8AAIrugZX//wAAiu6BmP//AACK7oGe//8AAIru gZ3//
wAAiu4BCAARAUMACIGZAAQAAIruABsAEAAMAAQABwACAAAsAAgAEAAGB
kgABAACK7oGjAAEAAIrugZwAAgAAiu6Blv//AACK7oGX//8AAIrugZT//wAAiu6Blf//
AACK7oGY//8AAIrugZ7//wAAiu6Bnf//AACK7g==
```

The UAC registration in Section 6.2 is in this message:

```
AAoA2EzAO88AAAAAAAwOQEBAGsAAAEpE2YTkwAAAAHGM2QBxjNkChPEE8QR
DAIPc2lwOmV4YW1wbGUuY29tAAAVc2lwOmFsaWNlQGV4YW1wbGUuY29tBTc2
eWhoFWY4MS1kNC1mNkBlGfTcGx1LmNvbQZjLXRyLTEAAQIAXQAAASkTZhUk
AAAAAcYzZArGM2QBE8QTxBEMAQDIAAAVc2lwOmFsaWNlQGV4YW1wbGUuY29t
BTc2eWhoFWY4MS1kNC1mNkBlGfTcGx1LmNvbQZjLXRyLTEA
```

The direct call in Section 6.3 is in this message:

```
AAoCGkzAPA8AAAAAAAwOQEBAlGAAAEpE2YTkwAAACDGM2QBywBxARPEE8QR
BQIYc2lwOmJvYkBiB2IxlMv4YW1wbGUubmV0E3NpcDpib2JAZXhhbXBsZS5u
```

ZXQAFXNpcDphbGljZUBleGFtcGx1LmNvbQU3NnloaBVMODItZDQtZjdAZXhhbXBsZS5jb20HYy0xLXh0NgABAgB5AAABKRNmGKoAAAAgywBxAcYzZAETxBPEEQUBALQTc2lwOmJvYkBleGFtcGx1Lm5ldAhiLWluNi1pdRVzaXA6YWxpY2VAZXhhbXBsZS5jb20FNzZ5aGgVZjgyLWQ0LWY3QGV4YW1wbGUuY29tB2MtMS14dDYAAQIAeQAAASkTZhz0AAAAIMsAcQHGM2QBE8QTxBEFAQDIE3NpcDpib2JAZXhhbXBsZS5uZXQIYi1pbjYtaXUVc2lwOmFsaWNlQGV4YW1wbGUuY29tBTc2eWhoFWY4MilkNC1mN0BleGFtcGx1LmNvbQdjLTeteHQ2AAEBAJAAAAEpE2YdCAAAACDGM2QBywBxARPEE8QRAQIYc2lwOmJvYkBib2IxLmV4YW1wbGUubmV0E3NpcDpib2JAZXhhbXBsZS5uZXQIYi1pbjYtaXUVc2lwOmFsaWNlQGV4YW1wbGUuY29tBTc2eWhoFWY4MilkNC1mN0BleGFtcGx1LmNvbQdjLTeteHQ2AA==

The downstream branch call in Section 6.4 is in this message:

AAoE4UzAPEoAAAAAAAwOQEBAH4AAAEpE2YTkWAAACvGM2QBxjNkChPEE8QRBQETc2lwOmJvYkBleGFtcGx1Lm5ldBNzaXA6Ym9iQGV4YW1wbGUubmV0ABVzaXA6YWxpY2VAZXhhbXBsZS5jb20EYWwtMRJ0ci04N2hAZXhhbXBsZS5jb20ABnMteC10cgECAGwAAAEpE2YUwQAAACvGM2QKxjNkARPEE8QRBQIAZBNzaXA6Ym9iQGV4YW1wbGUubmV0ABVzaXA6YWxpY2VAZXhhbXBsZS5jb20EYWwtMRJ0ci04N2hAZXhhbXBsZS5jb20ABnMteC10cgEBAIkAAAEpE2YYPgAAACvGM2QKxwBxARPEE8QRBQIYc2lwOmJvYkBib2IxLmV4YW1wbGUubmV0E3NpcDpib2JAZXhhbXBsZS5uZXQAFXNpcDphbGljZUBleGFtcGx1LmNvbQRhbc0xEnRyLTg3aEBleGFtcGx1LmNvbQZjLXgtDHIGcy14LXRyAQIAdgAAASkTZhlwAAAAK8sAcQHGM2QKE8QTxBEFAQBke3NpcDpib2JAZXhhbXBsZS5uZXQEYjEtMRVzaXA6YWxpY2VAZXhhbXBsZS5jb20EYWwtMRJ0ci04N2hAZXhhbXBsZS5jb20GYy14LXRyBnMteC10cgECAHYAAAEpE2YbyAAACvLAHEBxjNkChPEE8QRBQEAtBNzaXA6Ym9iQGV4YW1wbGUubmV0BGIXLTEVc2lwOmFsaWNlQGV4YW1wbGUuY29tBGFsLTESdHITODdoQGV4YW1wbGUuY29tBmMteC10cgZzLXgtDHIBAgB2AAABKRNmHJgAAAArxjNkCsYzZAETxBPEEQUCALQTc2lwOmJvYkBleGFtcGx1Lm5ldARiMS0xFXNpcDphbGljZUBleGFtcGx1LmNvbQRhbc0xEnRyLTg3aEBleGFtcGx1LmNvbQZjLXgtDHIGcy14LXRyAQIAdgAAASkTZiDwAAAAK8sAcQHGM2QKE8QTxBEFAQDIE3NpcDpib2JAZXhhbXBsZS5uZXQEYjEtMRVzaXA6YWxpY2VAZXhhbXBsZS5jb20EYWwtMRJ0ci04N2hAZXhhbXBsZS5jb20GYy14LXRyBnMteC10cgECAHYAAAEpE2YhpAAACvGM2QKxjNkARPEE8QRBQIAyBNzaXA6Ym9iQGV4YW1wbGUubmV0BGIXLTEVc2lwOmFsaWNlQGV4YW1wbGUuY29tBGFsLTESdHITODdoQGV4YW1wbGUuY29tBmMteC10cgZzLXgtDHIBAQCIAAABKRNmKKwAAAArxjNkAcYzZAOTxBPEEQEBE3NpcDpib2JAZXhhbXBsZS5uZXQTC2lwOmJvYkBleGFtcGx1Lm5ldARiMS0xFXNpcDphbGljZUBleGFtcGx1LmNvbQRhbc0xEnRyLTg3aEBleGFtcGx1LmNvbQZjLXgtDHIGcy14LXRyAQEAiAAAAASKTZiisAAAAK8YzZArLAHEBE8QTxBEBANzaXA6Ym9iQGV4YW1wbGUubmV0E3NpcDpib2JAZXhhbXBsZS5uZXQEYjEtMRVzaXA6YWxpY2VAZXhhbXBsZS5jb20EYWwtMRJ0ci04N2hAZXhhbXBsZS5jb20GYy14LXRyBnMteC10cg==

The forked call in Section 6.5 is in this message:

AAoHjEzAPF0AAAAAAAwOQEBAH4AAAEpE2YTkWAAACvGM2QBywBxyBPEE8QRBQETc2lwOmJvYkBleGFtcGx1Lm5ldBNzaXA6Ym9iQGV4YW1wbGUubmV0ABVzaXA6YWxpY2VAZXhhbXBsZS5jb20EYTETMRJ0ci04OGhAZXhhbXBsZS5jb20ABnMtMS10cgECAGwAAAEpE2YUwQAAACvLAHHIxjNkARPEE8QRBQIAZBNzaXA6

Ym9iQGV4YW1wbGUubmV0ABVzaXA6YWxpY2VAZXhhbXBsZS5jb20EYTETMRJ0  
ci04OGhAZXhhbXBsZS5jb20ABnMtMS10cgEBAIkAAAEpE2YYpgAAACvLAHHI  
ywBxARPEE8QRBQIYc2lwOmJvYkBiB2IXLmV4YW1wbGUubmV0E3NpcDpib2JA  
ZXhhbXBsZS5uZXQAFXNpcDphbGljZUBleGFtcGxlLmNvbQRhMS0xEnRyLTg4  
aEBleGFtcGxlLmNvbQZjLTETdHIGcy0xLXRyAQUALQAAASkTZhcAAAAK8sA  
ccggAQ24AAAAAAAAAAAAAAAAAAJE8QTxBEFAhhzaXA6Ym9iQGV4YjYjIuZXhhbXBs  
ZS5uZXQTC2lwOmJvYkBlEgfTcGxlLm5ldAAVc2lwOmFsaWNlQGV4YW1wbGUu  
Y29tBGExLTESdHITODhoQGV4YW1wbGUuY29tBmMtMi10cgZzLTETdHIBAgB2  
AAABKRNmG8gAAAArywBxAcsAccgTxBPeeQUBAGQTc2lwOmJvYkBlEgfTcGxl  
Lm5ldARiMS0xFXNpcDphbGljZUBleGFtcGxlLmNvbQRhMS0xEnRyLTg4aEB1  
eGFtcGxlLmNvbQZjLTETdHIGcy0xLXRyAQgAggAAASkTZhz0AAAAKyABDbgA  
AAAAAAAAAAAAAnLAHHIE8QTxBEFAQBKE3NpcDpib2JAZXhhbXBsZS5uZXQE  
YjItMhVzaXA6YWxpY2VAZXhhbXBsZS5jb20EYTETMRJ0ci04OGhAZXhhbXBs  
ZS5jb20GYy0yLXRyBnMtMS10cgEIAIIAAAEpE2YfTAAACsgAQ24AAAAAAAA  
AAAAAAAAJywBxyBPee8QRBQEAtBNzaXA6Ym9iQGV4YW1wbGUubmV0BGIyLTIV  
c2lwOmFsaWNlQGV4YW1wbGUuY29tBGExLTESdHITODhoQGV4YW1wbGUuY29t  
BmMtMi10cgZzLTETdHIBAgByAAABKRNmIG4AAAArywBxyMYzZAETxBPeeQUc  
ALQTc2lwOmJvYkBlEgfTcGxlLm5ldAAVc2lwOmFsaWNlQGV4YW1wbGUuY29t  
BGExLTESdHITODhoQGV4YW1wbGUuY29tBmMtMi10cgZzLTETdHIBAgB2AAAB  
KRNmIaQAAAArywBxyMYzZAETxBPeeQUcALQTc2lwOmJvYkBlEgfTcGxlLm5l  
dARiMS0xFXNpcDphbGljZUBleGFtcGxlLmNvbQRhMS0xEnRyLTg4aEBleGFt  
cGxlLmNvbQZjLTETdHIGcy0xLXRyAQIAdgAAASkTZiOYAAAAK8sAcQHLAHHI  
E8QTxBEFAQDIE3NpcDpib2JAZXhhbXBsZS5uZXQEYjEtMRVzaXA6YWxpY2VA  
ZXhhbXBsZS5jb20EYTETMRJ0ci04OGhAZXhhbXBsZS5jb20GYy0xLXRyBnMt  
MS10cgECAHYAAAEpE2YkYAAAAcVLAHHIXjNkARPEE8QRBQIAyBNzaXA6Ym9i  
QGV4YW1wbGUubmV0BGIxLTEVc2lwOmFsaWNlQGV4YW1wbGUuY29tBGExLTES  
dHITODhoQGV4YW1wbGUuY29tBmMtMS10cgZzLTETdHIBBQCVAABKRNmJSkA  
AArywBxyCABDbgAAAAAAAAAAAAAAAAAkTxBPeeQMCgHNpcDpib2JAYm9iMi5l  
eGFtcGxlLm5ldBNzaXA6Ym9iQGV4YW1wbGUubmV0ABVzaXA6YWxpY2VAZXhh  
bXBsZS5jb20EYTETMRJ0ci04OGhAZXhhbXBsZS5jb20GYy0yLXRyBnMtMS10  
cgEIAH4AAAEpE2YoPwAAACsgAQ24AAAAAAAAAAAAAAAAAAJywBxyBPee8QRBQEB  
5xNzaXA6Ym9iQGV4YW1wbGUubmV0ABVzaXA6YWxpY2VAZXhhbXBsZS5jb20E  
YTETMRJ0ci04OGhAZXhhbXBsZS5jb20GYy0yLXRyBnMtMS10cgEFAJUAAAEp  
E2YqDwAAACvLAHHIIAENuAAAAAAAAAAAAAAAAACRPEE8QRAQIYc2lwOmJvYkBi  
b2IyLmV4YW1wbGUubmV0E3NpcDpib2JAZXhhbXBsZS5uZXQAFXNpcDphbGlj  
ZUBleGFtcGxlLmNvbQRhMS0xEnRyLTg4aEBleGFtcGxlLmNvbQZjLTITdHIG  
cy0xLXRyAQgAfgAAASkTZiwAAAAKyABDbgAAAAAAAAAAAAAAAAAnLAHHIE8QT  
xBEDAQDIE3NpcDpib2JAZXhhbXBsZS5uZXQAFXNpcDphbGljZUBleGFtcGxl  
LmNvbQRhMS0xEnRyLTg4aEBleGFtcGxlLmNvbQZjLTITdHIGcy0xLXRy

## Authors' Addresses

Saverio Niccolini  
NEC Laboratories Europe, NEC Europe Ltd.  
Kurfuersten-Anlage 36  
Heidelberg 69115  
Germany

Phone: +49 (0) 6221 4342 118  
Email: [niccolini@neclab.eu](mailto:niccolini@neclab.eu)  
URI: <http://www.neclab.eu>

Benoit Claise  
Cisco Systems Inc.  
De Kleetlaan 6a b1  
Diegem, 1813  
Belgium

Phone: +32 2 704 5622  
Fax:  
Email: [bclaise@cisco.com](mailto:bclaise@cisco.com)  
URI:

Brian Trammell  
Swiss Federal Institute of Technology Zurich  
Gloriastrasse 35  
8092 Zurich  
Switzerland

Email: [trammell@tik.ee.ethz.ch](mailto:trammell@tik.ee.ethz.ch)

Hadriel Kaplan  
Acme Packet  
71 Third Ave.  
Burlington, MA 01803  
USA

Phone:  
Email: [hkaplan@acmepacket.com](mailto:hkaplan@acmepacket.com)



SIPCLF  
Internet-Draft  
Intended status: Standards Track  
Expires: April 28, 2011

G. Salgueiro  
Cisco Systems  
V. Gurbani  
Bell Labs, Alcatel-Lucent  
A. B. Roach  
Tekelec  
October 25, 2010

Indexed File Format for the Session Initiation Protocol (SIP)  
Common Log Format (CLF)  
draft-salgueiro-sipclf-indexed-ascii-02

## Abstract

The SIPCLF Workgroup has defined a common log format framework for Session Initiation Protocol (SIP) servers. This common log format mimics the wildly successful event logging mechanism 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 data model and provides an effective encoding scheme for all mandatory and optional fields that appear in a SIP CLF record.

## Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on April 28, 2011.

## Copyright Notice

Copyright (c) 2010 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 (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Terminology . . . . .	3
2. Introduction . . . . .	3
3. Format . . . . .	4
4. Example Record . . . . .	9
5. Text Tool Considerations . . . . .	11
6. Security Considerations . . . . .	12
7. Operational Guidance . . . . .	12
8. IANA Considerations . . . . .	12
9. Acknowledgements . . . . .	12
10. References . . . . .	12
10.1. Normative References . . . . .	12
10.2. Informative References . . . . .	12
Authors' Addresses . . . . .	13

## 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 RFC2119[1].

RFC3261[3] 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

The extensive list of benefits and the widespread adoption of the Apache Common Log Format (CLF) has prompted the development of a functionally equivalent event logging mechanism for the Session Initiation Protocol (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. 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 [2]. 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 data model presented in Section 8 of draft-ietf-sipclf-problem-statement [2]. This document defines 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.

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.

### 3. Format

Each SIP CLF record MUST consist of all the mandatory data model elements outlined in Section 8.1 of draft-ietf-sipclf-problem-statement [2] and each 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 ('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 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.

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

Client-Txn Pointer (Hex)			56 - 59
TLV Start Pointer (Hex)			60 - 63
0x0A			64 - 67
Timestamp			68 - 71
		0x2E	72 - 75
Fractional Seconds			76 - 79
Mandatory Fields			
0x09	Tag (Hex)		\ / > / \
Tag (cont)	0x2C	Length (Hex)	
Length (cont)		0x2C	
Value			
0x0A			

Repeated as many times as necessary

First, a 64-byte header indicates meta-data about the record.

Version (1 byte): 0x41 for this document; hexadecimal encoded.

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

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

u = received UDP message  
t = received TCP message  
l = received TLS message  
U = sent UDP message  
T = sent TCP message  
L = sent TLS message

Bytes 12 through 59 contain hexadecimal encoded pointers that point to the values of variable-length mandatory fields. 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 MUST be  $\geq 80$ . They point to the start of the corresponding value within the "Mandatory Fields" area.

When a given mandatory field is not applicable in the SIP CLF record (i.e. a particular entity is not able to log that field because it does not make sense for the role the entity is playing in the SIP ecosystem) then it MUST be encoded as a single ASCII dash (0x2D) and will consequently have a corresponding length field value of 1. 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.

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 ASCII 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, including the port number. The port number MUST be separated from the IP address by a single ':'.

Source IP address:port The IP address of the upstream client, including the port number over which the SIP message was received. The port number MUST be separated from the IP address by a single ':'.

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 in the From header field.

Whilst one may question the value of the From URI in light of RFC4474 [5], the From URI, nonetheless, imparts some information. For one, the From tag is important and, in the case of a REGISTER request, the From URI can provide information on whether this was a third-party registration or a first-party one.

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 The Common Log Format (CLF) for the Session Initiation Protocol (SIP) [2] 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 The Common Log Format (CLF) for the Session Initiation Protocol (SIP) [2] for usage.)

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

Timestamp (10 bytes): 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). Decimal encoded.

Fractional Seconds (6 bytes): Fractional seconds portion of the Timestamp field to millisecond accuracy. 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. If a given mandatory field is not present then it MUST be encoded as a horizontal dash ("-").

After the "Mandatory Fields" section, the OPTIONAL Tag/Length/Value groups appear zero or more times. These TLV groups allow SIP CLF implementers the flexibility to extend the logging capability of the indexed-ASCII representation beyond just the mandatory log elements. The location within the SIP CLF record is indicated by the "TLV Start Pointer" field. This "TLV Start Pointer" field MUST be set to 0x0000 if the OPTIONAL TLV groups are not implemented.

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 - Remote Host (mandatory in request)  
The DNS name of the IP address from which the message was received (if "Sent/Received flag" is set to "r"). The DNS name of the IP address to which the message is being sent (if "Sent/Received flag" is set to "s")

0x0002 - Authenticated User

Contains the user name by which the user has been authenticated

0x0003 - Complete SIP Message (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).

#### 4. Example Record

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



```
INVITE sip:192.168.217.74 SIP/2.0
To: <sip:192.168.217.74>
Call-ID: DL70dff590c1-1079051554@petermac.magor.local.
From: "PeterM" <sip:1001@petermac.magor.local.:5060>;tag=DL88360fa5fc;
      epid=0x34619b0
CSeq: 1 INVITE
Max-Forwards: 70
Via: SIP/2.0/TCP 192.168.217.117:5060;branch=z9hG4bK-1f6be070c4-DL
Contact: "1001" <sip:1001@192.168.217.117:5060>
Allow: INVITE,CANCEL,ACK,OPTIONS,INFO,SUBSCRIBE,NOTIFY,BYE,MESSAGE,
      UPDATE,REFER
Supported: replaces,norefersub
User-Agent: Dylogic Mirial 7.0.33
Content-Type: application/sdp
Content-Length: 418
```

```
v=0
o=1001 1456139204 0 IN IP4 192.168.217.117
s=-
i=Dylogic Mirial 7.0.33
c=IN IP4 192.168.217.117
b=AS:2048
t=0 0
m=audio 13756 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=x-mpdp:192.168.217.117:13756
m=video 13758 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=420015; max-mps=47520; max-fs=1584;
      max-dpb=7680
a=x-mpdp:192.168.217.117:13758
```

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 internet-draft conventions, this log entry has been split into seven 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.

A000120,Rou,0051005A005C006F0083009900AC00AE00D200DF010D01170000

```
0000000000.010  1 INVITE          -          sip:192.168.217.74
192.168.217.74:5060      192.168.217.117:56485
sip:192.168.217.74      -          sip:1001@petermac.magor.local.:5060
DL88360fa5fc      DL70dff590c1-1079051554@petermac.magor.local.
server-tx          client-tx
```

A Base64 encoded version of this log entry (without the changes required to format it for an internet-draft) is shown below:

begin-base64 644 clf\_record

```
QTAWMDEyMCxSb3UsMDA1MTAwNUeWMDVMDA2RjAwODMwMDk5MDEBBQzAwQUUwMEQyMDEBERjAxMEQw
MTE3MDAwMAowMDAwMDAwMDAwLjAxMAkxIElOVklURQktCXNpcDoxOTIuMTY4LjIxNy43NAkxOTIu
MTY4LjIxNy43ND01MDYwCTE5Mi4xNjguMjE3LjExNz01NjQ4NQ1zaXA6MTkyLjE2OC4yMTcuNzQJ
LQ1zaXA6MTAwMUBwZXRLcm1hYy5tYWdvci5sb2Nhbc46NTA2MA1ETDg4MzYwZmE1ZmMJREw3MGRm
ZjU5MGMxLTEwNzkwNTE1NTRAcGV0ZXJtYWwubWFnbn3IubG9jYWwuCXNlcnZlci10eAljbGllbnQt
dHgK
=====
```

## 5. 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]*)/`.

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.

## 6. Security Considerations

This document does not introduce any new security considerations beyond those in draft-ietf-sipclf-problem-statement [2].

## 7. 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.

## 8. IANA Considerations

This document does not require any considerations from IANA.

## 9. Acknowledgements

TBD

## 10. References

### 10.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [2] Gurbani, V., Burger, E., Anjali, T., Abdelnur, H., and O. Festor, "The Common Log Format (CLF) for the Session Initiation Protocol (SIP)", draft-ietf-sipclf-problem-statement-03 (work in progress), June 2010.

### 10.2. Informative References

- [3] 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.
- [4] Gurbani, V., Burger, E., Anjali, T., Abdelnur, H., and O. Festor, "The Common Log File (CLF) format for the Session

Initiation Protocol (SIP)", draft-gurbani-sipping-clf-01 (work in progress), March 2009.

- [5] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.

#### 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

