Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: July 27, 2014

P. Dawes
Vodafone Group

January 23, 2014

Solutions for Marking SIP Messages to be Logged
draft-dawes-insipid-logme-solutions-00

Abstract

SIP networks use signalling monitoring tools to diagnose user reported problem and for regression testing if network or user agent software is upgraded. As networks grow and become interconnected, including connection via transit networks, it becomes impractical to predict the path that SIP signalling will take between user agents, and therefore impractical to monitor SIP signalling end-to-end.

This draft describes potential solutions to meet requirements for adding an indicator to the SIP protocol which can be used to mark signalling as of interest to logging. Such marking will typically be applied as part of network testing controlled by the network operator and not used in regular user agent signalling. However, such marking can be carried end-to-end including the SIP user agents, even if a session originates and terminates in different networks.

Table of Contents

1. Introduction

If users experience problems with setting up sessions using SIP, their service provider needs to find out why by examining the SIP signalling. Also, if network or user agent software or hardware is upgraded regression testing is needed. Such diagnostics apply to a small proportion of network traffic and can apply end-to-end, even if signalling crosses several networks possibly belonging to several different network operators. It may not be possible to predict the path through those networks in advance, therefore a mechanism is needed to mark a session as being of interest to enable SIP entities along the signalling path to provide diagnostic logging. This draft describes potential solutions to meet the requirements for such a 'log me' marker for SIP signalling defined in draft-dawes-insipid-logme-reqs [I-D.dawes-insipid-logme-reqs].

2. Requirements Language

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].

3. Motivating Scenario

Signalling for SIP session setup can cross several networks, and these networks may not have common ownership and also may be in differrent countries. If a single operator wishes to perform regression testing or fault diagnosis end-to-end, the separate ownership of networks that carry the signalling and the explosion in the number of possible signalling paths through SIP entities from the originating to the terminating user make it impractical to pre-configure logging of an end-to-end SIP signalling of a session of interest.

The figure below shows an example of a signalling path through multiple networks.

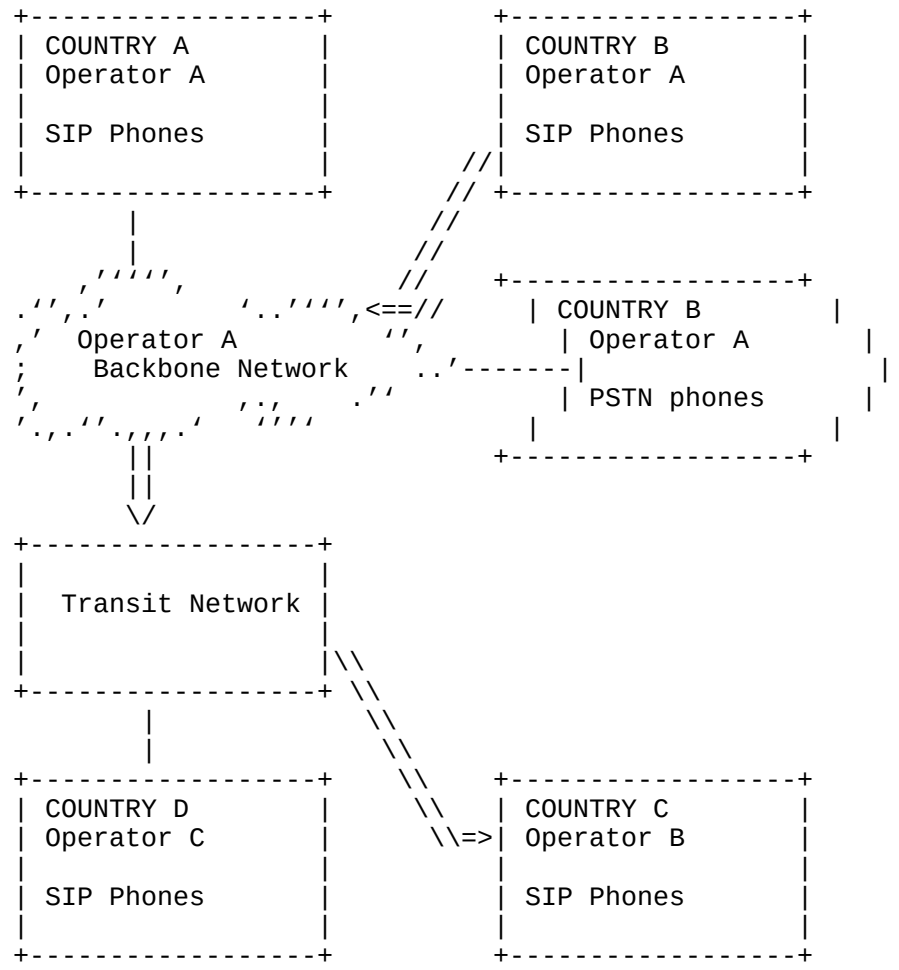


Figure 1: Example signalling path through multiple networks

4. Skeleton Diagnostic Procedure

The skeleton diagnostic procedure is as follows:

- The user's user agent is placed in debug mode. The user agent logs its own signalling and inserts a log me marker into SIP requests for session setup

- All SIP entities that the signalling traverses, from the first proxy the user agent connects to at the edge of the network to the destination user agent, can detect that the log me marker is

present and can log SIP requests and responses that contain the marker if configured to do so.

- Subsequent responses and requests in the same dialog are logged.
- Logging stops, either because the dialog has ended or because a 'stop event', typically expiry of a certain amount of time, occurred
- The user's user agent and any other SIP entity that has logged signalling sends logs to a server that is co-ordinating diagnostics.

5. Potential Solutions for Log Me Marking

This clause describes and compares potential solutions to the log-me requirements described in draft-dawes-insipid-logme-reqs [I-D.dawes-insipid-logme-reqs].

5.1. Functionality Common to all Solutions

5.1.1. Starting and Stopping log-me marking

A proxy or user agent needs to determine when it needs to log-me mark a SIP request or response. A user agent or proxy log-me marks a request or response for two reasons: either it is configured to do so or it has detected that a dialog is being log-me marked and maintains state to ensure that all requests and responses in the dialog are log-me marked. A regression test might be configured to log-me mark all SIP dialogs created during a given time period whereas a troubleshooting test might be configured to mark a dialog based on criteria specific to a reported fault. When configuration has caused a user agent or proxy to start log-me marking requests and responses, marking continues until the dialog ends.

5.1.2. Configuration for log-me marking

Configuration of a user agent or proxy to perform log-me marking can be done in any way that is convenient to the configured entity. For example, an XML file might be used to list conditions for starting and stopping based on time.

```
<start>09:00:00</start>
<stop>09:10:00</stop>
```

Figure 2: Simple example log-me configuration

Logging is on a per-dialog basis and individual logs are differentiated by their test identifier (test identifier is described in draft-dawes-insipid-logme-reqs [I-D.dawes-insipid-logme-reqs]). Therefore, an individual log for an individual dialog is closed when that dialog ends. Logging is typically done separately from regualar operation, which means that tests can be designed to be short enough to troubleshoot quickly and to limit the size of individual logs. If logging is configured so that everything is logged for a specified number of minutes then several separate dialogs might start and finish meaning that several logs may be generated, each one distinguished by its test identifier.

5.1.3. End Points of Logme Marking

Log-me marking is initiated on a dialog creating side controlled by configuration. The dialog terminating side detects an incoming log-me marker and reacts accordingly.

5.1.3.1. Originating and Terminating User Agent

In the simplest case, an originating user agent will insert a log-me marker in the dialog-creating SIP request and all subsequent SIP requests within that dialog. The log-me marker is carried to the terminating user agent and the terminating user agent echoes the log-me marker in responses. If the terminating user agent sends an in-dialog request on a dialog that is being log-me marked, it inserts a log-me marker and the originating user agent echoes the log-me marker in responses. This basic case suggests the following principles:

- The originating user agent is configured for debug
- The terminating user agent is not configured for debug and cannot initiate log-me marking.
- The originating user agent logs its own signalling and inserts a log me marker into the dialog-creating SIP request and subsequent in-dialog SIP requests.
- The terminating user agent can detect that a dialog is of interest to logging by the existence of a log-me marker in an incoming dialog-creating SIP request.
- The terminating user agent must echo a log-me marker in responses to a SIP request that included a log-me marker.
- If the terminating user agent has detected that a dialog is being log-me marked, it inserts a log-me marker in any in-dialog SIP requests that it sends.

5.1.3.2. Originating Edge Proxy and Terminating Edge Proxy

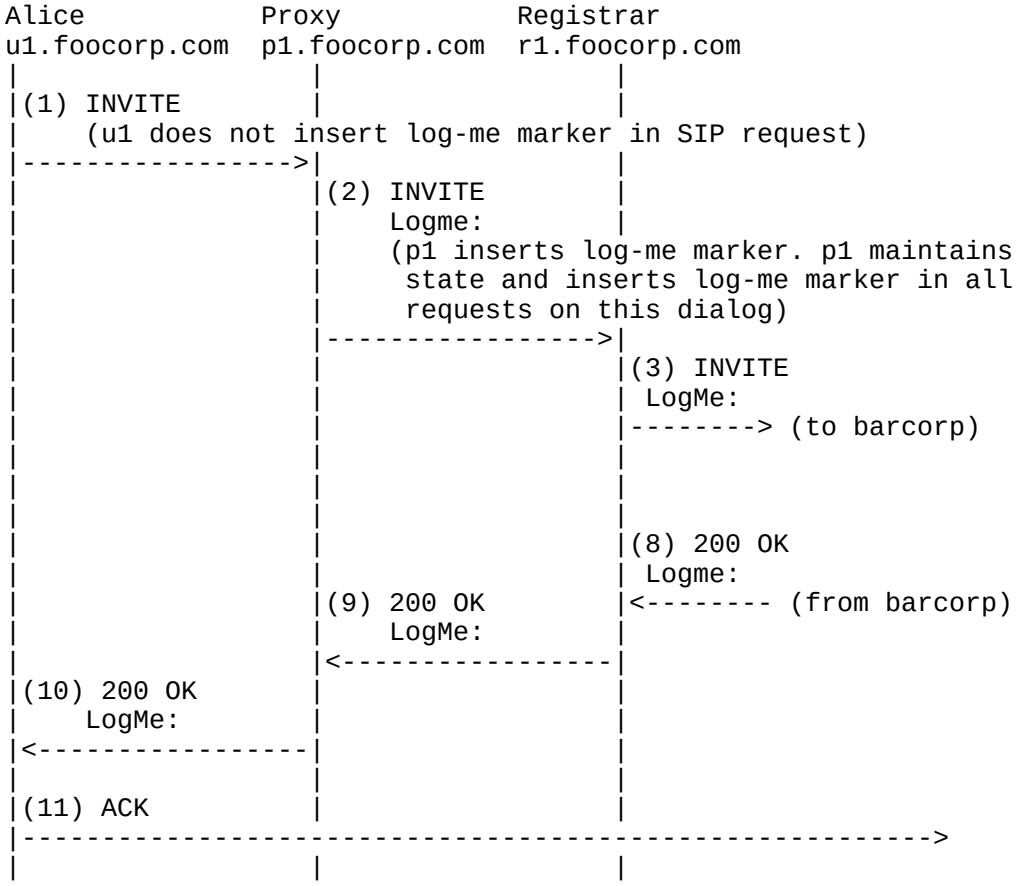
Some user agents might not support log-me marking. In order to test sessions involving such user agents, the log-me marker is inserted by edge proxies on the originating and terminating sides. The log-me marker is carried to the terminating user agent but the terminating user agent is not able to echo the log-me marker in responses to that request. Therefore the terminating edge proxy inserts a log-me marker in reponses to the request. Likewise, if the terminating user agent sends an in-dialog request, the terminating edge proxy inserts a log-me marker and the originating edge proxy echoes the log-me marker in responses to that request. This case suggests the following principles:

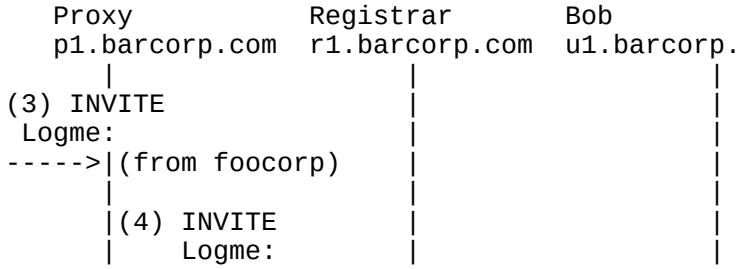
- The originating edge proxy is configured for debug.
- The terminating edge proxy is not configured for debug and cannot initiate log-me marking.
- The originating edge proxy logs its own signalling and inserts a log me marker into SIP requests for session setup.
- The terminating edge proxy can detect that a dialog is of interest to logging by the existence of a log-me marker in an incoming SIP request.
- The terminating edge proxy must echo a log-me marker in responses to a SIP request that included a log-me marker.
- If the terminating edge proxy has detected that a dialog is being log-me marked, it inserts a log-me marker in in-dialog SIP requests from the terminating user agent.
- The originating edge proxy echoes the log-me marker in responses to in-dialog requests received from the terminating side.

5.1.4. Maintaining State

If a proxy inserts a log-me marker in a SIP request (because a user agent did not) then it must ensure that a log-me marker is also inserted in responses to that request. A proxy on the terminating side that receives a SIP reqeust with a log-me marker may also ensure that responses to that requset contain a log-me marker by inserting one if the terminating user agent did not. Entities that perform this log-me marking or checking must maintain a record of which dialogs are being log-me marked.

In the figure below, the edge proxy in the originating network maintains state to ensure log-me marking of SIP requests and in the terminating network the registrar maintains state to ensure log-me marking of SIP responses. Such behaviour is useful for logging if end devices do not insert or echo a log-me marker.





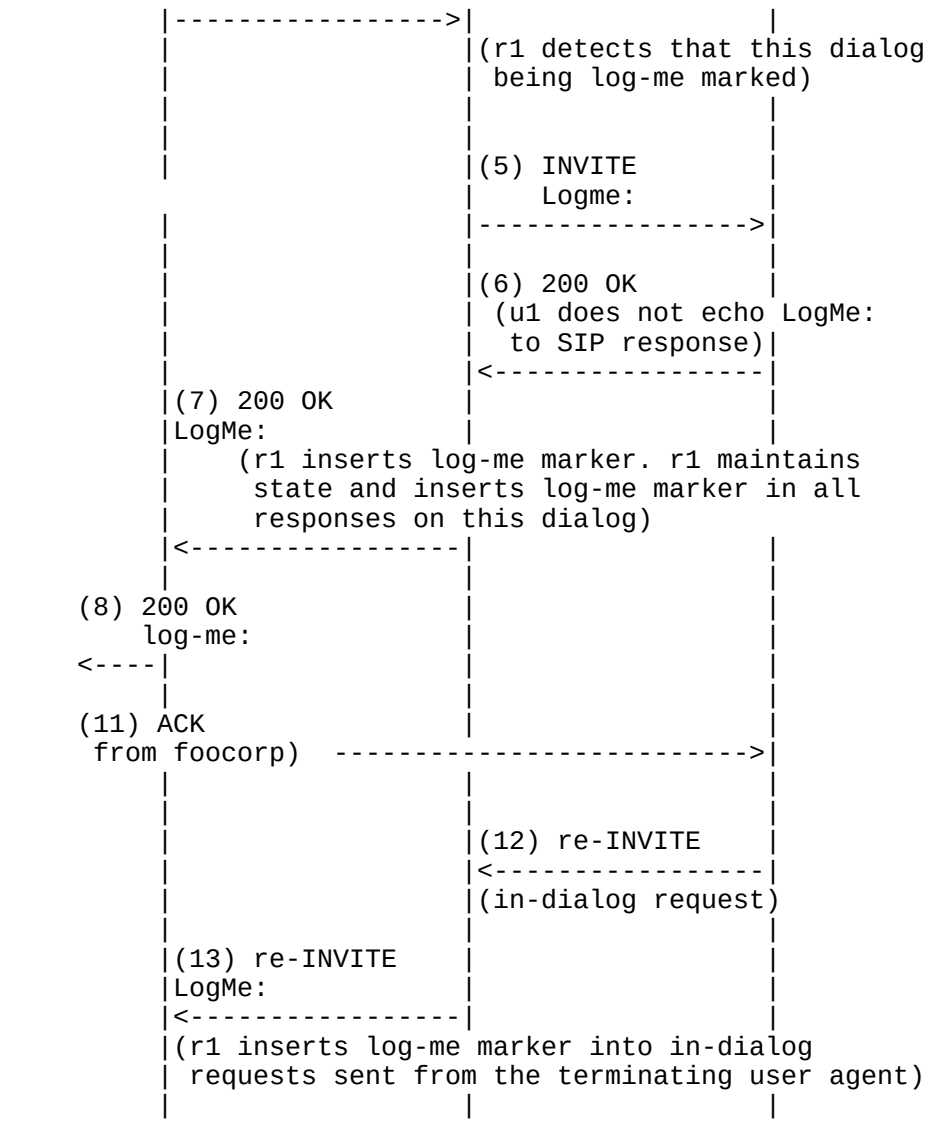


Figure 3: Maintaining state for log-me marking

5.1.5. Missing Log-me Marker in Dialog Being Logged

A terminating user agent or terminating edge proxy that has been echoing markers in responses for a given dialog might receive a SIP request that has not been log-me marked. Since log-me marking is done per dialog, this is an error. In such cases, the proxy should

consider log-me marking to have ended and not mark a response to the unmarked request, responses to subsequent requests in the dialog, or in-dialog requests sent from the terminating side. Similarly, log-me marking that begins mid-dialog is an error case and the terminating user agent or edge proxy must not log-me mark responses to the marked request, responses to subsequent requests in the dialog, or in-dialog requests from the terminating side.

5.1.6. Logging Multiple Simultaneous Dialogs

An originating or terminating user agent and SIP entities on the signaling path can log multiple SIP dialogs simultaneously, these dialogs can be differentiated by their test identifier.

5.1.7. Forked Requests and Back to Back User Agents

5.1.7.1. Propagating the Log-me Marker in Forked Requests

Log-me marking is copied into forked requests.

5.1.7.2. B2BUA processing of Log-me Marker

A B2BUA should act on the terminating side as described for a terminating user agent and therefore log marked incoming requests, echo log-me marking in responses to log-me marked requests, and log-me mark in-dialog SIP requests that it sends if the dialog is being log-me marked.

A B2BUA should act on the originating side as described for an originating user agent and therefore mark SIP requests if and only if configured to do so, and echo log-me marking in responses to in-dialog requests received from the terminating side.

5.1.8. Sending logs to a debug server

5.1.8.1. Protecting logs

A SIP entity that has logged information should encrypt it, such that it can be decrypted only by the debug server, before sending in order to protect the content of logs from a third party.

5.2. Solution A: Log-Me header field

5.2.1. Log-me Marker

A new SIP header field, e.g. 'LogMe:', is defined as the log-me marker. The LogMe header field is defined with one header field

parameter that contains a free-text name of the test case being performed.

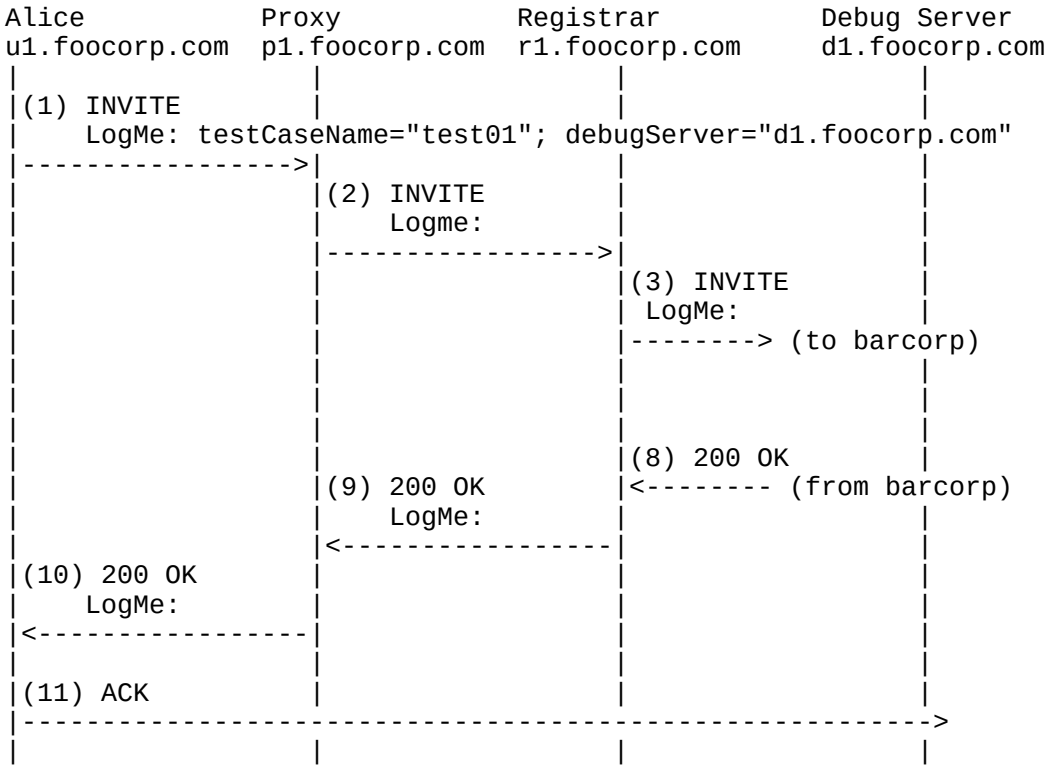
5.2.2. Identifying test cases

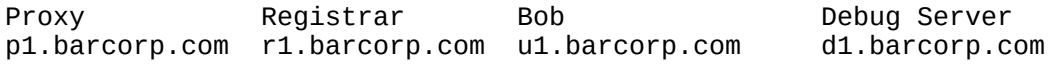
The new header field has a parameter that contains a free-text name of the test case being performed which acts as the test case identifier (test identifier is described in draft-dawes-insipid-logme-reqs [I-D.dawes-insipid-logme-reqs]).

5.2.3. Collecting logged information at a debug server

User agents and SIP proxies may send logged information to a debug server.

5.2.4. Examples





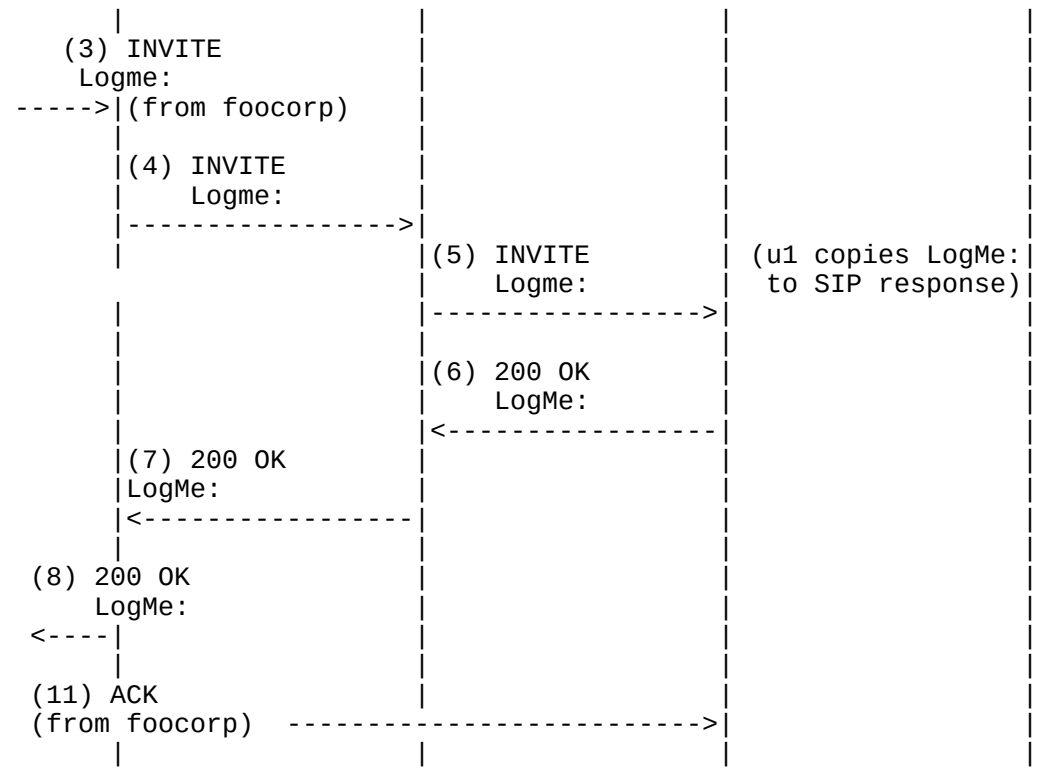


Figure 4: Signalling example for the LogMe header field solution

5.3. Solution B: New Value for purpose header field parameter in Call-Info:

5.3.1. Log-me Marker

A new value is defined for the purpose header field parameter used in Call-Info header field as the log-me marker.

The Call-Info: header field is defined in clause 20.9 of RFC 3261 [RFC3261].

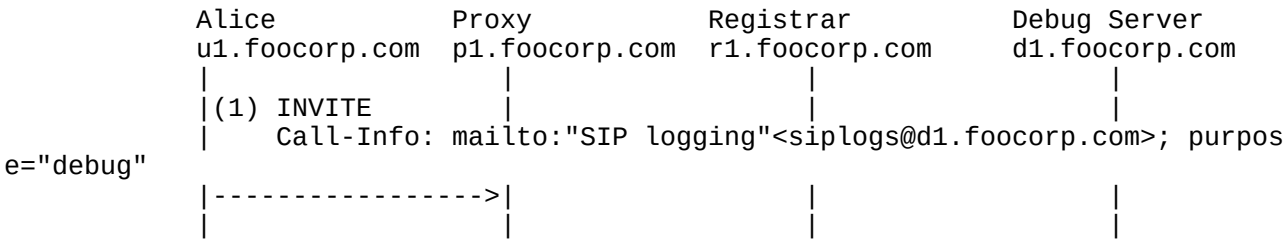


Figure 5: Signalling example for the Call-Info: purpose parameter solution

The Call-Info: header field can be included in methods INVITE, OPTIONS, REGISTER (Table 2: Summary of header fields, A--O in RFC 3261 [RFC3261] clause 20.1), INFO (RFC 6086 [RFC6086]), MESSAGE (RFC 3428 [RFC3428]), PUBLISH (RFC 3903 [RFC3903]), and UPDATE (RFC 3311 [RFC3311]), and in responses to those methods. Call-Info: header field cannot be included in methods NOTIFY, SUBSCRIBE, PRACK, or REFER.

5.3.2. Identifying test cases

The Call-Info: header field has no protocol element that can be used to indicate the test case name, therefore in this solution the test case is identified by the Session-ID header field (described in I-D .ietf-insipid-session-id-reqts [I-D.ietf-insipid-session-id-reqts]).

5.4. Solution C: New 'debug' header field parameter to be used in Session-ID header field

5.4.1. Log-me Marker

A new header field parameter called debug is defined to be used with the Session-ID header field (described in I-D.ietf-insipid-session-id-reqts [I-D.ietf-insipid-session-id-reqts]).

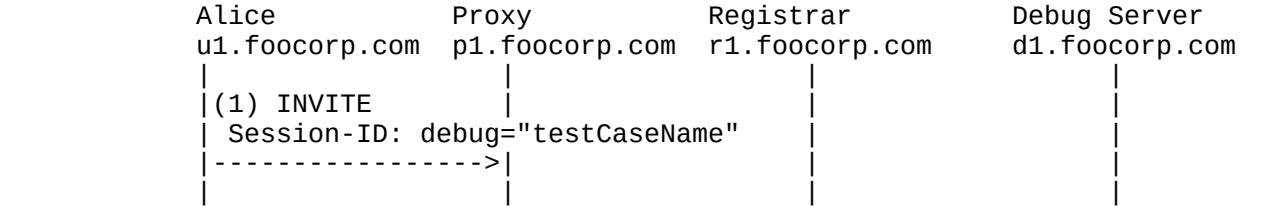


Figure 6: Signalling example for the Session-ID: header field parameter solution

5.4.2. Identifying test cases

In this solution the test case is identified by the Session-ID header field (described in I-D.ietf-insipid-session-id-reqts [I-D.ietf-insipid-session-id-reqts]).

5.5. Comparison of Potential Solutions

The table below summarizes the features of each potential solution. Other solutions are not excluded.

	Solution	Summary
A	Log-Me: header field	Specify a new SIP header field. Could be included in all SIP requests and responses. All behaviour including proxy handling in terms of add, delete, modify, and read, and which requests may or may not include the header field must be defined.
B	New value for the purpose parameter of the Call-Info header field e.g. "debug"	Rules for including, reading, modifying etc. are already defined by Call-Info. Call-Info cannot be inserted in all requests and responses, but can be included for the SIP methods of most interest to debugging and regression testing. No element to hold a test case name so test case is identified by the Session-ID header field.
C	New header field parameter for Session-ID header field e.g. debug	Might be viewed as a reason to remove the Session-ID header field, which would violate the Session-ID requirement: "REQ3: The solution must require that the identifier, if present, pass unchanged through SIP B2BUAs or other intermediaries." in I-D.ietf-insipid-session-id-reqts [I-D.ietf-insipid-session-id-reqts]

Table 1: Summary comparison of potential solutions

6. Security Considerations

All drafts are required to have a security considerations section. See RFC 3552 [RFC3552] for a guide.

6.1. Trust Domain

Since a log me marker may cause a SIP entity to log the SIP header and body of a request or response, the log me marker should be removed at a trust domain boundary. If a prior agreement to log sessions exists with the net hop network then the log me marker might not be removed.

6.2. Security Threats

6.2.1. Log-me marking

The log me marker is not sensitive information, although it will sometimes be inserted because a particular device is experiencing problems.

The presence of a log me marker will cause some SIP entities to log signalling. Therefore, this marker must be removed at the earliest opportunity if it has been incorrectly inserted.

Activating a debug mode affects the operation of a user agent, therefore it must be supplied by an authorized server to an authorized user agent, it must not be altered in transit, and it must not be readable by an unauthorized third party.

Logged signalling is privacy-sensitive data, therefore it must be passed to an authorized server, it must not be altered in transit, and it must not be readable by an unauthorized third party.

6.2.2. Debug server address

Log me marking may include the address of a debug server in the form of a URL. In order to prevent sending of logs to an unauthorised server a SIP entity that supports logging should authenticate the debug server, for example by referring to a statically configured white list of allowed destination domains.

6.2.3. Sending logged information

A SIP entity that has logged information should encrypt it, such that it can be decrypted only by the debug server, before sending it to a debug server in order to protect the content of logs from a third party.

7. References

7.1. Normative References

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

7.2. Informative References

[I-D.dawes-insipid-logme-reqs]
Dawes, P., "Requirements for Marking SIP Messages to be Logged", draft-dawes-insipid-logme-reqs-00 (work in progress), January 2014.

[I-D.ietf-insipid-session-id-reqts]
Jones, P., Salgueiro, G., Polk, J., Liess, L., and H. Kaplan, "Requirements for an End-to-End Session Identification in IP-Based Multimedia Communication Networks", draft-ietf-insipid-session-id-reqts-07 (work in progress), June 2013.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

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

[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.

[RFC3311] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.

[RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[RFC3903] Niemi, A., "Session Initiation Protocol (SIP) Extension for Event State Publication", RFC 3903, October 2004.

[RFC6086] Holmberg, C., Burger, E., and H. Kaplan, "Session Initiation Protocol (SIP) INFO Method and Package Framework", RFC 6086, January 2011.

Appendix A. Additional Stuff

This becomes an Appendix.

Author's Address

Peter Dawes
Vodafone Group
The Connection
Newbury, Berkshire RG14 2FN
UK

Phone: +44 7717 275009
Email: peter.dawes@vodafone.com