3GPP TSG SA WG3 Security — S3#15 **S3-000577**

12-14 September, 2000

Washington D.C., USA

SIP Working Group
Internet Draft
Document: <draft-dcsgroup-sip-state-02.txt>

W. Marshall
K. Ramakrishnan
AT&T

E. Miller
G. Russell
CableLabs

B. Beser
M. Mannette
K. Steinbrenner
3Com

D. Oran
F. Andreasen
Cisco

J. Pickens
Com21

P. Lalwaney
Nokia

J. Fellows
Motorola

D. Evans
Secure Cable Solutions

K. Kelly
NetSpeak

July, 2000

SIP Extensions for supporting Distributed Call State

http://www.ietf.org/ietf/1id-abstracts.txt

1. Abstract

This document describes an extension to the Session Initiation Protocol (SIP) that enables proxies to distribute call state to user agents. The state information can be returned to the proxy when the user agent requests a change in the characteristics of the active call. By providing the ability to distribute state to the user agents where it can be securely stored, proxy servers can remain stateless for the duration of the call. This mechanism allows a proxy server to provide services that depend on call state, while still being stateless.

2. Conventions used in this document

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 [2].

3. Introduction

In the Session Initiation Protocol (SIP) [4] proxies play the role of routing engines and delivery platforms for services. Many types of services require these proxies to retain call state. That is, these proxies know how to correlate SIP messages in order to reconstruct the state of calls that exist in the user agents. Unfortunately, maintaining call state presents problems. First, it introduces scalability problems when there are many user agents being served by a single proxy. Second, it makes failover and load balancing more complex, since once state is established in one proxy, subsequent signaling must return to the same proxy in order for proper service execution.

To achieve scalability when handling signaling messages from a large number of calls, SIP proxies must minimize the per call information that they need to maintain. One method of achieving this is for the proxy to transfer the state associated with a call to entities where the state is relevant. In addition, the proxy should be able to retrieve and update the call state information if the characteristics of the active call are changed.

The extension proposed in this document allows proxies to encapsulate any state information they desire into a header, called a State header, that is delivered to the user agents for a call. This information is reflected back in subsequent messages. This effectively allows proxies to store call state in user agents - behaving as SIP stateful proxies while still being stateless.

In this draft, we propose the following extension to SIP to support the distribution of call state:

1) A new general State header field that can be used to distribute call state information by the proxy to the UA during call setup or mid-call. The state information can also be encrypted, and contain an integrity check value, to guarantee detection of tampering by an untrusted UA.

If the UA wishes to change call characteristics, it passes the saved state information (which may be proxy encrypted and integrity protected) in a SIP INVITE request to its proxy server. The proxy is then able to perform the requested action, just as if the proxy had maintained the call state information itself. By using this mechanism, the proxies can offer the full range of services, yet remain stateless during the call.

The above mechanism for distributing state information is used in the Distributed call signaling (DCS) architecture [5].

2) A new option tag "state" is defined. This is to be used in the Supported header [5] by the initiating UA in its request to inform its proxy server that it understands and supports the behavior required by the State header. The responses would also include the Supported header with the option tag "state". In addition, proxy servers that transfer State to the UAS MUST also include a Require and a Proxy-Require header field with the option tag "state" if the proxy requires support for the extension.

4. Protocol Overview

Outlined below is an overview of the usage of the State header for distributing call state.

Consider a basic SIP INVITE-200 OK-ACK transaction. The UAC initiating the call sends an INVITE request to its proxy with the called party information. If the UAC supports the State header, the Supported header with the option tag "state" MUST be included in the request. The originating proxy locates the SIP proxy associated with the called party (referred to here as the terminating proxy) and forwards the INVITE to it. After the terminating proxy processes the INVITE, it has the information about the call being set up. The terminating proxy can pass this state information to the terminating/called UA in the State header. The State header includes a host value to identify the proxy that inserted the state token(s) that follows. In addition, the proxy MAY insert a Require and a Proxy-Require header field with the value "state" if it wishes the call to only be established if the State extension can be supported.

If the UAS supports the State extension, the State header along with the Supported header with an option tag of "state" is reflected back in the response. When the response to the INVITE (200 OK or the first non-100 1xx response) arrives at the originating proxy, the proxy has the complete call state information about the call being setup. When forwarding the response to the calling UA, the proxy includes this call state information in the State header.

The state information distribution described above between the proxy and the UA works for a network of proxies in the signaling path as well. If a proxy along the path wishes to distribute call state to the user agents, it adds a State header to the request (or the response). The State header includes a host value by which the proxy can identify itself followed by its state token(s) and any State header(s) inserted by other proxies.

The UAS that receives the State header(s) stores the headers and associates them with the call-leg.

The rules for when and how the stored state information is returned by the UA to the proxy are discussed in detail in the next section.

5. SIP Header Extension and Option Tag for Distributing Call State

If the State header is to be used to distribute state in a call, the UAC initiating the call MUST include the Supported header defined in [6] with the option tag "state" in the initial INVITE request.

UAS's receiving the Supported header with the value "state" MUST include the Supported header with an option tag of "state" in responses if they are capable of processing the State header extension.

A proxy in the signaling path MUST insert a Require and a Proxy-Require header with an option tag of "state" if it inserts a State header in the request or response.

5.1 State Header Syntax

The State header contains any information a proxy would like returned to it in subsequent messaging from the UA's for the same call leg. This might include information for support of mid-call features, billing information, etc. It is RECOMMENDED that this information be protected by an integrity check mechanism. This allows the proxy to reliably and securely store state information in the client that may be needed for subsequent feature invocation.

The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in RFC-2234 [3].

```
     State           = "State" ":" 1#(host ";" state-token
                               *(";" state-token))
     state-token     =  token ["=" (*token | quoted-string)]
```

The host field identifies the proxy that inserted the state information.

State headers may be nested. In this case, a proxy in the signaling path takes the State header(s) it received in the incoming signaling message (previous host; token form), possibly adds any state-tokens of its own, and generates a single new State header. The hostname in the nested State header identifies the proxy that performed the nesting.

Multiple State headers MAY be present in a request (or response). In addition, the syntax allows for a proxy to insert multiple tokens in the header.

The state token is a proxy-defined encoding of a structure containing multiple pieces of information needed by the proxy to perform various call features. The structure is returned from the UA to its proxy for call services that affect the current call.

The following defines the entry for the State header of Table 5 in RFC 2543.

	Where	enc	e-e	ACK	BYE	CAN	INV	OPT	REG
State	gc	n	h	o	o	o	o	o	o

6 Detailed Protocol Semantics

The protocol semantics for a UAC, a UAS and the proxy are addressed in this section.

6.1 UAC behavior

The rules at the UAC for processing State headers are listed below:

1. A UAC supporting this extension MUST include a Supported header field with an option tag of "state" in the initial INVITE and all subsequent requests and responses.

2. The UAC MUST save the received State header(s) along with the From, To, Call-ID and tags associated with the To and From header fields for the duration of the call.

3. On a subsequent request, the UAC includes the State header(s) in the request if the From, To (including ones with From and To reversed), Call-ID and the tags on the From and To match those associated with the saved State header(s) and Request-URI matches the hostname of the saved State header(s). If Route header is

present, the UAC also includes State headers that have hostname matching a component of the Route header.

4. Additional rules MAY be defined by other extensions that specify when a State header is to be included in a request. An example of this would be extensions that handle call transfers and other features that would specify State header processing at the UAC.

5. When a call leg ends, the UAC MAY delete all saved State headers associated with the call leg.

6.2 UAS behavior

The rules at the UAS for processing State headers are listed below:

1. A UAS that supports this extension MUST include a Supported header with the token value "state" in all responses.

2. The UAS MUST save the received State header(s) along with the From, To, Call-ID and tags associated with the To and From header fields for the duration of the call, or until a new request with the State is received.

3. In all non-100 responses to all requests, the UAS MUST include the State header(s) received in that request, and a Supported: "state" header.

6.3 Proxy Behavior

To support this extension, the Proxy MUST perform the following functions:

1. A State header that is received in a request or response, with a hostname other than the proxy's, MUST be passed on.

2. A Proxy that hides Via headers in a request MUST nest all the State headers received in the request. Further, the proxy MUST restore these State headers when that nested State header is received in a request or response.

3. A proxy that hides Record-Route headers in a request MUST nest all the State headers received in that request. Further, the proxy MUST restore these State headers when that nested State header is received in a request or response.

4. Requirements on a proxy that hides Record-Route headers in a response, or that hides Route headers, is for further study.

In addition, a proxy MAY do the following to utilize the capability offered by this extension:

1. A State header received in a request or response with the hostname matching the proxy MAY be discarded.

2. A proxy MAY generate one or more State headers, and include it (or them) in any request or response. A proxy that generates State headers MUST insert a "Require: state" header, and a "Proxy-Require: state" header, in the request if not already present.

3. A proxy MAY nest all, or any subset, of the State headers received in a request or response. A proxy that nests State headers MUST restore these State headers when that nested State header is received in a request or response.

6.4 Example of use

The following example illustrates the distribution of state during call setup and issues associated with concatenation and encryption of State headers. UAC and UAS refer to the originating and terminating User Agent for the call. P1 is the proxy associated with UAC and P2 is the proxy associated with UAS. eP1{*} refers to the state token encrypted by P1.

UAC -> P1 -> P2 -> UAS

```
UAC->P1:        invite
                Supported: state

P1->P2:         invite
                State:P1;state=eP1{"cached translation
                                of UAS's number"}
                Supported: state
                Require: state
```

In this example, P2 formulates a single State header by combining the State header received from the previous proxy(ies).

```
P2->UAS:        invite
                State:P2;state=eP2{"hunt group ID,
                        billing ID,P1;state=eP1{"cached
                        translation of UAS's number"}"}
                Supported: state
                Require: state
```

UAS saves the above state information received from its proxy P2 for the duration of the call.

```
UAS->P2:        response
                State:P2; state=eP2{"hunt group ID,
                        billing ID,P1;state=eP1{"cached
                        translation of UAS's number"}"}
                Supported: state
```

As P2 combined all State headers into one when sending the INVITE to the UAS, it is responsible for restoring the State headers as received in the INVITE before forwarding the response to P1 with its updated State header.

```
     P2->P1:         response
                     State:P2;state=eP2{"hunt group ID,
                             billing ID"},P1;state=eP1{"cached
                             translation of UAS's number"}
                     Supported: state

     P1->UAC:        response
                     state:P1;state=eP1{"billing ID,
                             cached translation of  UAS's
                     number, P2;state=eP2{"hunt group ID,
                                     billing ID"}"}
                     Supported: state
```

UAC saves the state information received from P1 for the duration of the call.

When the call begins, state at UAC is:

```
     State:P1;state=eP1{"billing ID, cached translation of UAS's
     number", P2;state=eP2{"hunt group ID, billing ID"}"}
```

State at UAS is:

```
     State:P2;state=eP2{"hunt group ID, billing ID,P1;state=eP1{"
     cached translation of UAS's number"}"}
```

Note that the state information for the call at the UAC and UAS is different. Proxies therefore need to be aware of the direction from which they receive the State header. This may be information included in the state token or may be deduced from other headers in the message.

7 State Header and HTTP Cookie/Pcookie Comparison

The State header field discussed in this section differs from the HTTP1.1 Cookies as described in [7]. In a general sense, both transfer state between the server and the client. HTTP uses the Cookie for "state" management, or as a handle to pass session context change from server to client where the server is the other endpoint of the session. Cookies typically persist across sessions. On the other hand, the State header is used to transfer current call state from a proxy or intermediate network proxies to the UAC and the UAS. The state header can be considered to be a handle to request a change in the active/current session by the endpoint from its proxy. In addition, there are no attribute value pairs associated with the State header as there are in the Cookie mechanism.

8 Security Considerations

If the clients/endpoints are considered untrusted entities, the proxy must encrypt the State header and include an integrity check with the State header information. In addition, the proxy is responsible for verifying the contents and integrity of the State header returned by the client as discussed in this document.

9 Notice Regarding Intellectual Property Rights

AT&T may seek patent or other intellectual property protection for some or all of the technologies disclosed in the document. If any standards arising from this disclosure are or become protected by one or more patents assigned to AT&T, AT&T intends to disclose those patents and license them on reasonable and non-discriminatory terms. Future revisions of this draft may contain additional information regarding specific intellectual property protection sought or received.

3COM may seek patent or other intellectual property protection for some or all of the technologies disclosed in the document. If any standards arising from this disclosure are or become protected by one or more patents assigned to 3COM, 3COM intends to disclose those patents and license them on reasonable and non-discriminatory terms. Future revisions of this draft may contain additional information regarding specific intellectual property protection sought or received.

10 References

1. Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

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

3. Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997

4. M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg,"SIP: session initiation protocol," Request for Comments (Proposed Standard) 2543, Internet Engineering Task Force, Mar. 1999.

5. Marshall, W. et. al, "Architectural Considerations for Providing Carrier Class Telephony Services Utilizing SIP-based Distributed Call Control Mechanisms", Internet Draft, Internet Engineering Task Force, draft-dcsgroup-sip-arch-02, June 2000, Work In Progress

6. J. Rosenberg and H. Schulzrinne, "The SIP Supported Header", draft-ietf-sip-serverfeatures-02.txt, September 2000.

7.   Kristol, D. and Montulli, L., "HTTP State Management Mechanism", RFC 2109, February 1997. See current working draft <draft-ietf-http-state-man-mec-12.txt> modified by the same authors based on field implementation feedback.

11 Acknowledgements

The Distributed Call Signaling work in the PacketCable project is the work of a large number of people, representing many different companies. The authors would like to recognize and thank the following for their assistance: John Wheeler, Motorola; David Boardman, Daniel Paul, Arris Interactive; Bill Blum, Jon Fellows, Jay Strater, Jeff Ollis, Clive Holborow, Motorola; Doug Newlin, Guido Schuster, Ikhlaq Sidhu, 3Com; Jiri Matousek, Bay Networks; Farzi Khazai, Nortel; John Chapman, Bill Guckel, Michael Ramalho, Cisco; Chuck Kalmanek, Doug Nortz, John Lawser, James Cheng, Tung-Hai Hsiao, Partho Mishra, AT&T; Telcordia Technologies; and Lucent Cable Communications.

Many thanks to Jonathan Rosenberg for extensive comments on this draft.

12 Author's Addresses

Bill Marshall
AT&T
Florham Park, NJ 07932
Email: wtm@research.att.com

K. K. Ramakrishnan
AT&T
Florham Park, NJ 07932
Email: kkrama@research.att.com

Ed Miller
CableLabs
Louisville, CO 80027
Email: E.Miller@Cablelabs.com

Glenn Russell
CableLabs
Louisville, CO 80027
Email: G.Russell@Cablelabs.com

Burcak Beser
3Com
Rolling Meadows, IL 60008
Email: Burcak_Beser@3com.com

Mike Mannette

3Com
Rolling Meadows, IL 60008
Email: Michael_Mannette@3com.com

Kurt Steinbrenner
3Com
Rolling Meadows, IL 60008
Email: Kurt_Steinbrenner@3com.com

Dave Oran
Cisco
Acton, MA 01720
Email: oran@cisco.com

Flemming Andreasen
Cisco
Edison, NJ
Email: fandreas@cisco.com

John Pickens
Com21
San Jose, CA
Email: jpickens@com21.com

Poornima Lalwaney
Nokia
San Diego, CA 92121
Email: poornima.lalwaney@nokia.com

Jon Fellows
Motorola
San Diego, CA 92121
Email: jfellows@gi.com

Doc Evans
Secure Cable Solutions
Westminster, CO 30120
Email: drevans@securecable.com

Keith Kelly
NetSpeak
Boca Raton, FL 33587
Email: keith@netspeak.com

Full Copyright Statement

Expiration Date: This memo is filed as <draft-dcsgroup-sip-state-02.txt>, and expires January 2001.