Robust Header Compression
Internet-Draft
Intended status: Standards Track
Expires: March 19, 2008

C. Bormann
Universitaet Bremen TZI
Z. Liu
Nokia Research Center
R. Price
EADS Defence and Security Systems Limited
G. Camarillo, Ed.
Ericsson

September 20, 2007

Applying Signaling Compression (SigComp) to the Session Initiation Protocol (SIP)
draft-ietf-rohc-sigcomp-sip-08.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

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

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

This Internet-Draft will expire on March 19, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

This document describes some specifics that apply when Signaling

Compression (SigComp) is applied to the Session Initiation Protocol (SIP), such as default minimum values of SigComp parameters, compartment and state management, and a few issues on SigComp over TCP. Any implementation of SigComp for use with SIP must conform to this document and SigComp, and in addition support the SIP and Session Description Protocol (SDP) static dictionary.

Table of Contents

1. Introduction

SigComp [RFC3320] is a solution for compressing messages generated by application protocols. Although its primary driver is to compress SIP [RFC3261] messages, the solution itself has been intentionally designed to be application agnostic so that it can be applied to any application protocol; this is denoted as ANY/SigComp. Consequently, many application dependent specifics are left out of the base standard. It is intended that a separate specification is used to describe those specifics when SigComp is applied to a particular application protocol.

This document binds SigComp and SIP; this is denoted as SIP/SigComp.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Compliance with this Specification

Any SigComp implementation that is used for the compression of SIP messages MUST conform to this document, as well as to [RFC3320]. Additionally, it must support the SIP/SDP static dictionary, as specified in [RFC3485], and the mechanism for discovering SigComp support at the SIP layer, as specified in [RFC3486].

4. Minimum Values of SigComp Parameters for SIP/SigComp

In order to support a wide range of capabilities among endpoints implementing SigComp, SigComp defines a few parameters to describe SigComp behavior (see section 3.3 of [RFC3320]). For each parameter, [RFC3320] specifies a minimum value that any SigComp endpoint MUST support for ANY/SigComp. Those minimum values were determined with the consideration of all imaginable devices in which SigComp may be implemented. Scalability was also considered as a key factor.

However, some of the minimum values specified in [RFC3320] are too small to allow good performance for SIP message compression. Therefore, they are increased for SIP/SigComp as specified in the following sections. For completeness, those parameters that are the same for SIP/SigComp as they are for ANY/SigComp are also listed.

The new minimum values are specific to SIP/SigComp and, thus, do not

apply to any other application protocols. A SIP/SigComp endpoint MAY offer additional resources over and above the minimum values specified in this document if available; these resources can be advertised to remote endpoints as described in section 9.4.9 of [RFC3320].

4.1. decompression_memory_size (DMS) for SIP/SigComp

Minimum value for ANY/SigComp: 2048 bytes, as specified in section 3.3.1 of [RFC3320].

Minimum value for SIP/SigComp: 8192 bytes.

Reason: a DMS of 2048 bytes is too small for SIP message compression as it seriously limits the compression ratio and even makes compression impossible for certain messages. For example, the condition set by [RFC3320] for SigComp over UDP means: C + 2*B + R + 2*S + 128 < DMS (each term is described below). Therefore, if DMs is too small, at least one of C, B, R, or S will be severely restricted. On the other hand, DMS is memory that is only temporarily needed during decompression of a SigComp message (the memory can be reclaimed when the message has been decompressed). Therefore, a requirement of 8 KB should not cause any problem for an endpoint that already implements SIP, SigComp, and applications that use SIP.

C size of compressed application message, depending on R
B size of bytecode. Note: two copies -- one as part of the SigComp message and one in UDVM (Universal Decompressor Virtual Machine) memory.
R size of circular buffer in UDVM memory
S any additional state uploaded other than that created from the content of the circular buffer at the end of decompression (similar to B, two copies of S are needed)
128 the smallest address in UDVM memory to copy bytecode to

4.2. state_memory_size (SMS) for SIP/SigComp

Minimum value for ANY/SigComp: 0 (zero) bytes, as specified in section 3.3.1 of [RFC3320].

Minimum value for SIP/SigComp: 2048 bytes.

Reason: a non-zero SMS allows an endpoint to upload a state in the first SIP message sent to a remote endpoint without the uncertainty of whether the remote endpoint will have enough memory to store such a state. A non-zero SMS obviously requires the SIP/SigComp implementation to keep state. Based on the observation that there is little gain from stateless SigComp compression, the assumption is

that purely stateless SIP implementations are unlikely to provide a SigComp function. Stateful implementations should have little problem to keep 2K additional state for each compartment (see Section 9).

Note: SMS is a parameter that applies to each individual compartment. An endpoint MAY offer different SMS values for different compartments as long as the SMS value is not less than 2048 bytes.

4.3. cycles_per_bit (CPB) for SIP/SigComp

Minimum value for ANY/SigComp: 16, as specified in section 3.3.1 of [RFC3320].

Minimum value for SIP/SigComp: 16 (same as above)

4.4. SigComp_version (SV) for SIP/SigComp

For ANY/SigComp: 0x01, as specified in section 3.3.2 of [RFC3320].

For SIP/SigComp: >= 0x02 (at least SigComp + NACK)

Note that this implies that the provisions of [RFC4077] apply. That is, decompression failures result in SigComp NACK messages sent back to the originating compressor. It also implies that the compressor need not make use of the methods detailed in Section 2.4 of [RFC4077] (Detecting Support for NACK); for example, it can use optimistic compression methods right from the outset.

4.5. locally available state (LAS) for SIP/SigComp

Minimum LAS for ANY/SigComp: none, see section 3.3.3 of [RFC3320].

Minimum LAS for SIP/SigComp: the SIP/SDP static dictionary as defined in [RFC3485].

Note that, since support for the static SIP/SDP dictionary is mandatory, it does not need to be advertised.

5. Delimiting SIP Messages and SigComp Messages on the Same Port

In order to limit the number of ports required by a SigComp-aware endpoint, it is possible to allow both SigComp messages and 'vanilla' SIP messages (i.e. uncompressed SIP messages with no SigComp header) to arrive on the same port.

For a message-based transport such as UDP or SCTP, distinguishing

between SigComp and non-SigComp messages can be done per message. The receiving endpoint checks the first octet of the UDP/SCTP payload to determine whether the message has been compressed using SigComp. If the MSBs (Most Significant Bits) of the octet are "11111" then the message is considered to be a SigComp message and is parsed as per [RFC3320]. If the MSBs of the octet take any other value, then the message is assumed to be an uncompressed SIP message, and is passed directly to the application with no further effect on the SigComp layer.

For a stream-based transport such as TCP, distinguishing between SigComp and non-SigComp messages has to be done per connection. The receiving endpoint checks the first octet of the TCP data stream to determine whether the stream has been compressed using SigComp. If the MSBs of the octet are "11111" then the stream is considered to contain SigComp messages and is parsed as per [RFC3320]. If the MSBs of the octet take any other value, then the stream is assumed to contain uncompressed SIP messages, and is passed directly to the application with no further effect on the SigComp layer. Note that SigComp message delimiters MUST NOT be used if the stream contains uncompressed SIP messages.

Applications MUST NOT mix SIP messages and SigComp messages on a single TCP connection. If the TCP connection is used to carry SigComp messages then all messages sent over the connection MUST have a SigComp header and be delimited by the use of 0xFFFF as described in [RFC3320].

Section 11 of [RFC4896] details a simple set of bytecodes, intended to be "well-known", that implement a null decompression algorithm. These bytecodes effectively allow SigComp peers to send selected SigComp messages with uncompressed data. If a SIP implementation has reason to send both compressed and uncompressed SIP messages on a single TCP connection, the compressor can be instructed to use these bytecodes to send uncompressed SIP messages that are also valid SigComp messages.

6. Continuous Mode over TCP

Continuous Mode is a special feature of SigComp, which is designed to improve the overall compression ratio for long-lived connections. Its use requires pre-agreement between the SigComp compressor and decompressor. Continuous mode is not used with SIP/SigComp.

Reason: continuous mode requires the transport itself to provide a certain level of protection against denial of service attacks. TCP alone is not considered to provide enough protection.

7. Too Large SIP Messages

SigComp does not support the compression of messages larger than 64k. Therefore, if a SIP application sending compressed SIP messages to another SIP application over a transport connection (e.g., a TCP connection) needs to send a SIP message larger than 64k, the SIP application MUST NOT send the message over the same TCP connection. The SIP application SHOULD send the message over a different transport connection (to do this, the SIP application may need to establish a new transport connection).

8. SIP Retransmissions

When SIP messages are retransmitted, they need to be re-compressed, taking into account any SigComp states that may have been created or invalidated since the previous transmission. Implementations MUST NOT cache the result of compressing the message and retransmit such a cached result.

The reason for this behavior is that it is impossible to know whether the failure causing the retransmission occurred on the message being retransmitted or on the response to that message. If the response was lost, any state changes effected by the first instance of the retransmitted message would already have taken place. If these state changes removed a state that the previously-transmitted message relied upon, then retransmission of the same compressed message would lead to a decompression failure.

Note that a SIP retransmission may be caused by the original message or its response being lost by a decompression failure. In this case, a NACK will have been sent by the decompressor to the compressor, which may use the information in this NACK message to adjust its compression parameters. Note that, on an unreliable transport, such a NACK message may still be lost, so if a compressor used some form of optimistic compression it MAY want to switch to a method less likely to cause any form of decompression failure when compressing a SIP retransmission.

9. Compartment and State Management for SIP/SigComp

An application exchanging compressed traffic with a remote application has a compartment that contains state information needed to compress outgoing messages and to decompress incoming messages. To increase the compression efficiency, the application must assign distinct compartments to distinct remote applications.

9.1. Remote Application Identification

SIP/SigComp applications identify remote applications by their SIP/SigComp identifiers. Each SIP/SigComp application MUST have a SIP/SigComp identifier URN (Uniform Resource Name) that uniquely identifies the application. Usage of a URN provides a persistent and unique name for the SIP/SigComp identifier. It also provides an easy way to guarantee uniqueness. This URN MUST be persistent as long as the application stores compartment state related to other SIP/SigComp applications.

A SIP/Sigcomp application SHOULD use a UUID (Universally Unique IDentifier) URN as its SIP/SigComp identifier, due to the difficulties in equality comparisons for other kinds of URNs. The UUID URN [RFC4122] allows for non-centralized computation of a URN based on time, unique names (such as a MAC address), or a random number generator. If a URN scheme other than UUID is used, the URN MUST be selected such that the application can be certain that no other SIP/SigComp application would choose the same URN value.

Note that the definition of SIP/SigComp identifier is similar to the definition of instance identifier in [I-D.ietf-sip-outbound]. One difference is that instance identifiers are only required to be unique within their AoR (Address of Record) while SIP/SigComp identifiers are required to be globally unique.

Even if instance identifiers are only required to be unique within their AoR, devices may choose to generate globally unique instance identifiers. A device with a globally unique instance identifier SHOULD use its instance identifier as its SIP/SigComp identifier.

> Using the same value for an entity's instance and SIP/SigComp identifiers improves the compression ratio of header fields that carry both identifiers (e.g., a Contact header field in a REGISTER request).

Server farms that share SIP/SigComp state across servers MUST use the same SIP/SigComp identifier for all their servers.

SIP/SigComp identifiers are carried in the 'sigcomp-id' SIP URI (Uniform Resource Identifier) or Via header field parameter. The 'sigcomp-id' SIP URI parameter is a 'uri-parameter', as defined by the SIP ABNF (Augmented Backus-Naur Form, Section 25.1 of [RFC3261]). The following is its ABNF [RFC4234]:

```
uri-sip-sigcomp-id = "sigcomp-id=" 1*paramchar
```

The SIP URI 'sigcomp-id' parameter MUST contain a URN [RFC2141].

The Via 'sigcomp-id' parameter is a 'via-extension', as defined by the SIP ABNF (Section 25.1 of [RFC3261]). The following is its ABNF [RFC4234]:

```
via-sip-sigcomp-id = "sigcomp-id" EQUAL
                LDQUOT *( qdtext / quoted-pair ) RDQUOT
```

The Via 'sigcomp-id' parameter MUST contain a URN [RFC2141].

The following is an example of a 'sigcomp-id' SIP URI parameter:

```
sigcomp-id=urn:uuid:0C67446E-F1A1-11D9-94D3-000A95A0E128
```

The following is an example of a Via header field with a 'sigcomp-id' parameter:

```
Via: SIP/2.0/UDP server1.example.com:5060
   ;branch=z9hG4bK87a7
   ;comp=sigcomp
   ;sigcomp-id="urn:uuid:0C67446E-F1A1-11D9-94D3-000A95A0E128"
```

The following is an example of a REGISTER request that carries 'sigcomp-id' parameters in a Via entry and in the Contact header field. Additionally, it also carries a '+sip.instance' Contact header field parameter.

```
REGISTER sip:example.net SIP/2.0
Via: SIP/2.0/UDP 192.0.2.247:2078;branch=z9hG4bK-et736vsjirav;
  rport;sigcomp-id="urn:uuid:2e5fdc76-00be-4314-8202-1116fa82a473"
From: "Joe User" <sip:2145550500@example.net>;tag=6to4gh7t5j
To:  "Joe User" <sip:2145550500@example.net>
Call-ID: 3c26700c1adb-lu1lz5ri5orr
CSeq: 215196 REGISTER
Max-Forwards: 70
Contact: <sip:2145550500@192.0.2.247:2078;
  sigcomp-id=urn:uuid:2e5fdc76-00be-4314-8202-1116fa82a473>;
  q=1.0; expires=3600;
  +sip.instance="<urn:uuid:2e5fdc76-00be-4314-8202-1116fa82a473>"
Content-Length: 0
```

SIP messages are matched with remote application identifiers as follows.

Outgoing requests: the remote application identifier is the SIP/SigComp identifier of the URI to which the request is sent. If the URI does not contain a SIP/SigComp identifier, the remote application identifier is the IP address plus port of the datagram carrying the request for connection-less transport protocols, and

the transport connection (e.g., a TCP connection) carrying the request for connection-oriented transport protocols (this is to support legacy SIP/SigComp applications).

Incoming responses: the remote application identifier is the same as that of the previously-sent request that initiated the transaction to which the response belongs.

Incoming requests: the remote application identifier is the SIP/SigComp identifier of the top-most Via entry. If the Via header field does not contain a SIP/SigComp identifier, the remote application identifier is the source IP address plus port of the datagram carrying the request for connection-less transport protocols, and the transport connection (e.g., a TCP connection) carrying the request for connection-oriented transport protocols (this is to support legacy SIP/SigComp applications).

Outgoing responses: the remote application identifier is the same as that of the previously-received request that initiated the transaction to which the response belongs. Note that, due to standard SIP Via header field processing, this identifier will be present in the top-most Via entry in such responses (as long as it was present in the top-most Via entry of the previously-received request).

A SIP/SigComp application placing its URI with the 'comp=sigcomp' parameter in a header field MUST add a 'sigcomp-id' parameter with its SIP/SigComp identifier to that URI.

A SIP/SigComp application generating its own Via entry containing the 'comp=sigcomp' parameter MUST add a 'sigcomp-id' parameter with its SIP/SigComp identifier to that Via entry.

A given remote application identifier is mapped to a particular SigComp compartment ID following the rules given in Section 9.3.

9.2. Identifier Comparison Rules

Equality comparisons between SIP/SigComp identifiers are performed using the rules for URN equality that are specific to the scheme in the URN. If the element performing the comparisons does not understand the URN scheme, it performs the comparisons using the lexical equality rules defined in RFC 2141 [RFC2141]. Lexical equality may result in two URNs being considered unequal when they are actually equal. In this specific usage of URNs, the only element which provides the URN is the SIP/SigComp application identified by that URN. As a result, the SIP/SigComp application SHOULD provide lexically equivalent URNs in each registration it generates. This is

likely to be normal behavior in any case; applications are not likely to modify the value of their SIP/SigComp identifiers so that they remain functionally equivalent yet lexicographically different from previous identifiers.

9.3. Compartment Opening and Closure

SIP applications need to know when to open a new compartment and when to close it. The lifetime of SIP/SigComp compartments is linked to registration state. Compartments are opened at SIP registration time and are typically closed when the registration expires or is canceled.

> Note that linking the lifetime of SIP/SigComp compartments to registration state limits the applicability of this specification. In particular, SIP user agents that do not register but, for example, only handle PUBLISH or SUBSCRIBE/NOTIFY transactions are not able create SIP/SigComp compartments following this specification. Previous revisions of this specification also defined compartments valid during a SIP transaction or a SIP dialog. Those compartments covered all possible SIP entities, including those that do not handle REGISTER transactions. However, it was decided to eliminate those types of compartments because the complexity they introduced (e.g., edge proxy servers were required to keep dialog state) was higher than the benefits they brought in most deployment scenarios.

Usually, any states created during the lifetime of a compartment will be "logically" deleted when the compartment is closed. As described in section 6.2 of [RFC3320], a logical deletion can become a physical deletion only when no compartment continues to exist that created the (same) state.

A SigComp endpoint may offer to keep a state created upon request from a SigComp peer endpoint beyond the default lifetime of a compartment (i.e., beyond the duration of its associated registration). This may be used to improve compression efficiency of subsequent SIP messages generated by the same remote application at the SigComp peer endpoint. To indicate that such state will continue to be available, the SigComp endpoint can inform its peer SigComp endpoint by announcing the (partial) state ID in the returned SigComp parameters at the end of the registration that was supposed to limit the lifetime of the SigComp state. That signals the state will be maintained. The mandatory support for the SigComp Negative Acknowledgement (NACK) Mechanism [RFC4077] in SIP/SigComp ensures that it is possible to recover from synchronization errors regarding compartment lifetimes.

As an operational concern, bugs in the compartment management implementation are likely to lead to sporadic, hard to diagnose failures. Decompressors may therefore want to cache old state and, if still available, allow access while logging diagnostic information. Both compressors and decompressors use the SigComp Negative Acknowledgement (NACK) Mechanism [RFC4077] to recover from situations where such old state may no longer be available.

A REGISTER transaction causes an application to open a new compartment to be valid for the duration of the registration established by the REGISTER transaction.

A SIP application that needs to send a compressed SIP REGISTER (i.e., a user agent generating a REGISTER or a proxy server relaying one to its next hop) SHOULD open a compartment for the request's remote application identifier. A SIP application that receives a compressed SIP REGISTER (i.e., the registrar or a proxy relaying the REGISTER to its next-hop) SHOULD open a compartment for the request's remote application identifier.

These compartments MAY be closed if the REGISTER request is responded with a non-2xx final response, or when the registration expires or is canceled. However, applications MAY also choose to keep these compartments open for a longer period of time, as discussed previously. For a given successful registration, applications SHOULD NOT close their associated compartments until the registration is over.

> A SIP network can be configured so that regular SIP traffic to and from a user agent traverses a different set of proxies than the initial REGISTER transaction. The path the REGISTER transaction follows is typically determined by configuration data. The path subsequent requests traverse is determined by the Path [RFC3327] and the Service-Route [RFC3308] header fields in the REGISTER transaction and by the Record-Route and the Route header fields in dialog-creating transactions. Previous revisions of this document supported the use of different paths for different types of traffic. However, for simplicity reasons, this document now assumes that networks using compression are configured so that subsequent requests follow the same path as the initial REGISTER transaction. Section 10 provides network administrators with recommendations so that they can configure they networks properly.

If, following the rules above, a SIP application is supposed to open a compartment for a remote application identifier for which it already has a compartment (e.g., the SIP application registers towards a second registrar using the same edge proxy server as for its registration towards its first registrar), the SIP application

MUST use the already existing compartment. That is, the SIP application MUST NOT open a new compartment.

9.4. Lack of a Compartment

The use of stateless compression (i.e., compression without a compartment) is not typically worthwhile and may even result in message expansion. Therefore, if a SIP application does not have a compartment for a message it needs to send, it MAY choose not to compress it even in the presence of the comp=sigcomp parameter. Section Section 5 describes how a SIP application can send compressed and uncompressed messages over the same TCP connection. Note that RFC 3486 [RFC3486] states the following:

> "If the next-hop URI contains the parameter comp=sigcomp, the client SHOULD compress the request using SigComp"

Experience since RFC 3486 [RFC3486] was written has shown that stateless compression is, in most cases, not worthwhile. That is why now it is not recommended to use it any longer.

10. Recommendations for Network Administrators

Network administrators can configure their networks so that the compression efficiency achieved is increased. The following recommendations help network administrators perform their task.

For a given user agent, the route sets for incoming requests (created by a Path header field) and for outgoing requests (created by a Service-Route header field) are typically the same. However, registrars can, if they wish, insert proxies in the latter route that do not appear in the former route and vice versa. It is RECOMMENDED that registrars are configured so that proxies performing SigComp compression appear in both routes.

The routes described previously apply to requests sent outside a dialog. Requests inside a dialog follow a route constructed using Record-Route header fields. It is RECOMMENDED that the proxies performing SigComp that are in the route for requests outside a dialog are configured to place themselves (by inserting themselves in the Record-Route header fields) in the routes used for requests inside dialogs.

When a user agent's registration expires, proxy servers performing compression may close their associated SIP/SigComp compartment. If the user agent is involved in a dialog that was established before the registration expired, subsequent requests within the dialog may

not be compressed any longer. In order to avoid this situation, it is RECOMMENDED that user agents are registered as long as they are involved in a dialog.

11. Private Agreements

SIP/SigComp implementations that are subject to private agreements MAY deviate from this specification, if the private agreements unambiguously specify so. Plausible candidates for such deviations include:

- Minimum values (Section 4).
- Use of continuous mode (Section 6).
- Compartment definition (Section 9).

12. Backwards Compatibility

SigComp has a number of parameters that can be configured per endpoint. This document specifies a profile for SigComp when used for SIP compression that further constrains the range that some of these parameters may take. Examples of this are Decompressor Memory Size, State Memory Size, and SigComp Version (support for NACK). Additionally, this document specifies how SIP/SigComp applications should perform compartment mapping.

When this document was written, there already were a few existing SIP/SigComp deployments. The rules in this document have been designed to maximize interoperability with those legacy SIP/SigComp implementations. Nevertheless, implementers should be aware that legacy SIP/SigComp implementations may not conform to this specification. Examples of problems with legacy applications would be smaller DMS than mandated in this document, lack of NACK support, or a different compartment mapping.

13. Interactions with TLS

Endpoints exchanging SIP traffic over a TLS [RFC4346] connection can use the compression provided by TLS. Two endpoints exchanging SIP/SigComp traffic over a TLS connection that provides compression need to first compress the SIP messages using SigComp and, then, pass them to the TLS layer, which will compress them again. When receiving data, the processing order is reversed.

However, compressing messages twice this way does not typically bring significant gains. Once a message is compressed using SigComp, TLS

is not usually able to compress it further. Therefore, TLS will normally only be able to compress SigComp code sent between compressor and decompressor. Since the gain of having SigComp code compressed should be in most cases minimal, it is NOT RECOMMENDED using TLS compression when SigComp compression is being used.

14. Example

Figure 1 shows an example message flow where the user agent and the outbound proxy exchange compressed SIP traffic. Compressed messages are marked with a (c).

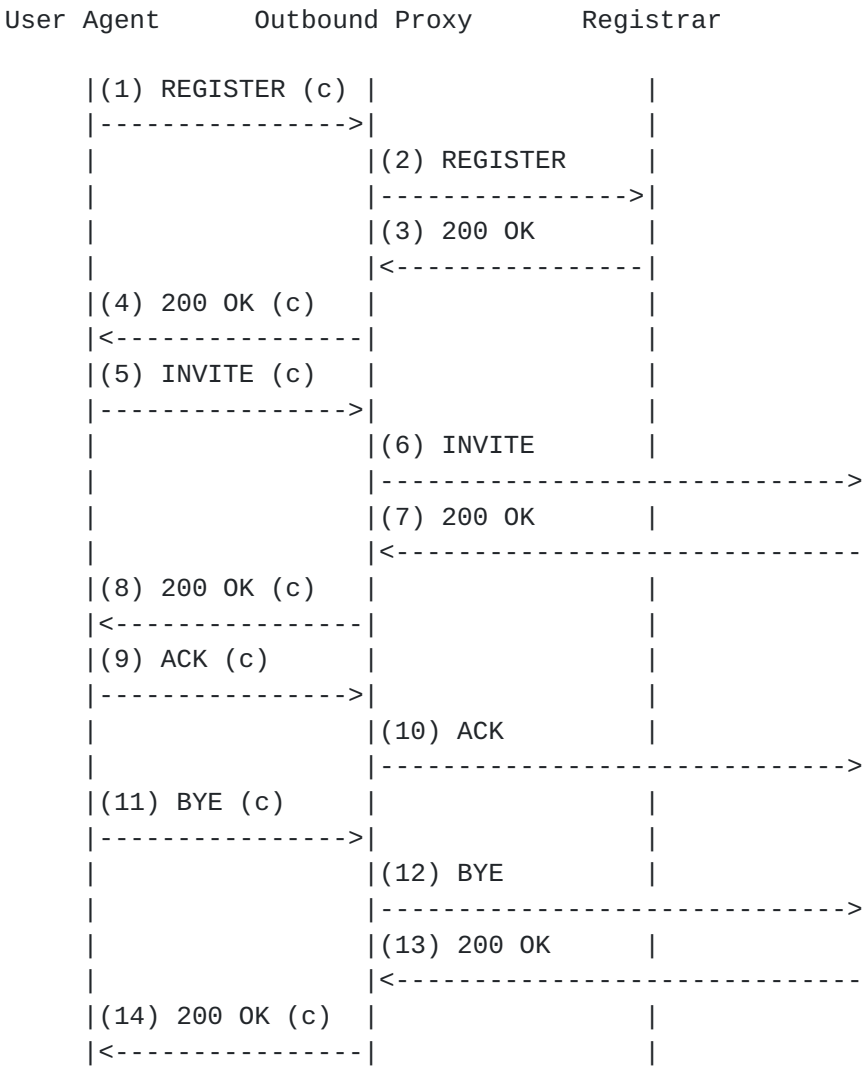

Figure 1: Example message flow

The user agent in Figure 1 is initially configured (e.g., using the SIP configuration framework [I-D.ietf-sipping-config-framework]) with the URI of its outbound proxy. That URI contains the outbound proxy's SIP/SigComp identifier, referred to as 'Outbound-id', in a 'sigcomp-id' parameter.

When the user agent sends an initial REGISTER request (1) to the outbound proxy's URI, the user agent opens a new compartment for 'Outbound-id'. This compartment will be valid, at least, for the duration of the registration.

On receiving this REGISTER request (1), the outbound proxy opens a new compartment for the SIP/SigComp identifier that appears in the 'sigcomp-id' parameter of the top-most Via entry. This identifier, which is the user agent's SIP/SigComp identifier, is referred to as 'UA-id'. The compartment opened by the outbound proxy will be valid, at least, for the duration of the registration. The outbound proxy adds Path header field with its own URI, which contains the 'Outbound-id' SIP/SigComp identifier, to the REGISTER request and relays it to the registrar (2).

When the registrar receives the REGISTER request (2), it constructs the route future incoming requests (to the user agent) will follow using the Contact and the Path header fields. Future incoming requests will traverse the outbound proxy before reaching the user agent.

The registrar also constructs the route future outgoing requests (from the user agent) will follow and places it in a Service-Route header field in a 200 (OK) response (3). Future outgoing requests will always traverse the outbound proxy. The registrar has ensured that the outbound proxy performing compression handles both incoming and outgoing requests.

When the outbound proxy receives a 200 (OK) response (3), it inspects the top-most Via entry. This entry's SIP/SigComp identifier 'UA-id' matches that of the compartment created before. Therefore, the outbound proxy uses that compartment to compress it and relay it to the user agent.

On receiving the 200 (OK) response (4), the user agent stores the Service-Route header field in order to use it to send future outgoing requests. The Service-Route header field contains the outbound proxy's URI, which contains the 'Outbound-id' SIP/SigComp identifier.

At a later point, the user agent needs to send an INVITE request (5). According to the Service-Route header field received previously, the user agent sends the INVITE request (5) to the outbound proxy's URI.

Since this URI's SIP/SigComp identifier 'Outbound-id' matches that of the compartment created before, this compartment is used to compress the INVITE request.

On receiving the INVITE request (5), the outbound proxy Record Routes and relays the INVITE request (6) forward. The outbound proxy Record Routes to ensure that all SIP messages related to this new dialog are routed through the outbound proxy.

Finally the dialog is terminated by a BYE transaction (11) that also traverses the outbound proxy.

15. Security Considerations

The same security considerations as described in [RFC3320] apply to this document. Note that keeping SigComp states longer than the duration of a SIP dialog should not pose new security risks because the state has been allowed to be created in the first place.

16. IANA Considerations

The IANA is requested to register the 'sigcomp-id' Via header field parameter, which is defined in Section 9.1, under the Header Field Parameters and Parameter Values subregistry within the SIP Parameters registry:

Header Field	Parameter Name	Predefined Values	Reference
Via	sigcomp-id	No	[RFCxxxx]

The IANA is requested to register the 'sigcomp-id' SIP URI parameter, which is defined in Section 9.1, under the SIP/SIPS URI Parameters subregistry within the SIP Parameters registry:

Parameter Name	Predefined Values	Reference
sigcomp-id	No	[RFCxxxx]

Note to the RFC Editor: please, substitute RFCxxxx with the RFC number this document will get.

17. Acknowledgements

The authors would like to thank the following people for their comments and suggestions: Jan Christoffersson, Joerg Ott, Mark West, Pekka Pessi, Robert Sugar, Jonathan Rosenberg, Robert Sparks, Juergen Schoenwaelder, and Tuukka Karvonen. Abigail Surtees and Adam Roach performed thorough reviews of this document.

18. References

18.1. Normative References

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

[RFC2141] Moats, R., "URN Syntax", RFC 2141, May 1997.

[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.

[RFC3308] Calhoun, P., Luo, W., McPherson, D., and K. Peirce, "Layer Two Tunneling Protocol (L2TP) Differentiated Services Extension", RFC 3308, November 2002.

[RFC3320] Price, R., Bormann, C., Christoffersson, J., Hannu, H., Liu, Z., and J. Rosenberg, "Signaling Compression (SigComp)", RFC 3320, January 2003.

[RFC3327] Willis, D. and B. Hoeneisen, "Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts", RFC 3327, December 2002.

[RFC3485] Garcia-Martin, M., Bormann, C., Ott, J., Price, R., and A. Roach, "The Session Initiation Protocol (SIP) and Session Description Protocol (SDP) Static Dictionary for Signaling Compression (SigComp)", RFC 3485, February 2003.

[RFC3486] Camarillo, G., "Compressing the Session Initiation Protocol (SIP)", RFC 3486, February 2003.

[RFC4077] Roach, A., "A Negative Acknowledgement Mechanism for Signaling Compression", RFC 4077, May 2005.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122,

July 2005.

[RFC4234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[RFC4896] Surtees, A., West, M., and A. Roach, "Signaling Compression (SigComp) Corrections and Clarifications", RFC 4896, June 2007.

18.2. Informative References

[I-D.ietf-sipping-config-framework]
Petrie, D. and S. Channabasappa, "A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-12 (work in progress), June 2007.

[I-D.ietf-sip-outbound]
Jennings, C. and R. Mahy, "Managing Client Initiated Connections in the Session Initiation Protocol (SIP)", draft-ietf-sip-outbound-08 (work in progress), March 2007.

Authors' Addresses

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28334
Germany

Phone: +49 421 218 7024
Fax: +49 421 218 7000
Email: cabo@tzi.org

Zhigang Liu
Nokia Research Center
955 Page Mill Road
Palo Alto, CA 94304
USA

Phone: +1 650 796 4578
Email: zhigang.c.liu@nokia.com

Richard Price
EADS Defence and Security Systems Limited
Meadows Road
Queensway Meadows
Newport, Gwent NP19 4SS

Phone: +44 (0)1633 637874
Email: richard.price@eads.com

Gonzalo Camarillo (editor)
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Gonzalo.Camarillo@ericsson.com

Full Copyright Statement

Copyright (C) The IETF Trust (2007).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).