Network Working Group
Internet Draft
Intended status: Standards Track
Expires: August 20, 2013

P. Jones
C. Pearce
J. Polk
G. Salgueiro
Cisco Systems

February 20, 2013

End-to-End Session Identification in IP-Based Multimedia Communication Networks
draft-jones-insipid-session-id-02.txt

Abstract

This document describes an end-to-end Session Identifier for use in IP-based Multimedia Communication systems that enables endpoints, intermediate devices, and management systems to identify a session end-to-end, associate multiple endpoints with a given multipoint conference, track communication sessions when they are redirected, and associate one or more media flows with a given communication session.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

IP-based multimedia communication systems like SIP [1] and H.323 [2] have the concept of a "call identifier" that is globally unique. The identifier is intended to represent an end-to-end communication session from the originating device to the terminating device. Such an identifier is useful for troubleshooting, session tracking, and so forth.

Unfortunately, there are a number of factors that contribute to the fact that the current call identifiers defined in SIP and H.323 are not suitable for end-to-end session identification. A fundamental issue in protocol interworking is the fact that the syntax for the call identifier in SIP and H.323 is different between the two protocols. This important fact makes it impossible for call identifiers to be exchanged end-to-end when a network utilizes one or more session protocols.

Another reason why the current call identifiers are not suitable to identify the session end-to-end is that in real-world deployments devices like session border controllers often change the session signaling as it passes through the device, including the value of the call identifier. While this is deliberate and useful, it makes it very difficult to track sessions end-to-end.

This draft presents a new identifier, referred to as the Session Identifier, or "Session ID", and associated syntax intended to overcome the issues that exist with the currently defined call identifiers. The proposal in this document attempts to comply with the requirements specified in Error! Reference source not found.. This proposal also has capabilities not mentioned in [5], shown in call flows in section 10. Additionally, this proposal attempts to account for a previous, proprietary version of a SIP Session ID header, proposing a backwards compatibility of sorts, described in section 11.

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 [3] when they appear in ALL CAPS. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

3. Session Identifier Requirements and Use Cases

Requirements and Use Cases for the end-to-end Session Identifier can be found in a separate memo titled "Requirements for an End-to-End Session Identification in IP-Based Multimedia Communication Networks" Error! Reference source not found..

4. Constructing the Session Identifier

The Session Identifier is comprised of two RFC 4122 defined UUIDs [4], with each UUID representing one of the endpoints participating in the session. The SIP user agent (UA) initially transmitting the SIP request will create a UUID and transmit that to the ultimate destination UA. Likewise, the responding UA will create a UUID and

transmit that to the first UA. These two distinct UUIDs form what is referred to as the Session Identifier and is represented in this document in set notation of the form {A,B}, where A is UUID value from the UA transmitting a message and B is the UUID value from the intended recipient of the message, i.e., not an intermediary server along the signaling path. The set {A,B} is equal to the set {B,A}, and thus both represent the same Session Identifier.

In the case where only one UUID is known, such as when a UA first initiates a SIP request, the Session ID would be {A}, where "A" represents the single UUID value transmitted.

Since SIP sessions are subject to any number of service interactions, SIP INVITE messages might be forked as sessions are established, and since conferences might be established or expanded with endpoints calling in or the conference focus calling out, the construction of the Session Identifier from a set of UUIDs is important.

To understand this better, consider that a UA participating in a communication session might be replaced with another, such as the case where two "legs" of a call are joined together by a PBX. Suppose that UA A and UA B both call UA C. Further suppose that UA C uses a local PBX function to join the call between itself and UA A with the call between itself and UA B. This merged call needs to be identified and identification of such sessions is natural and easily traceable when utilizing UUID values assigned by each entity in the communication session.

In the case of forking, UA A might send an INVITE that gets forked to five different UAs, as an example. Until one UA returns a 200 OK to the initial INVITE, a means of identifying each of these separate communication sessions is needed and allowing the set of {A, B1}, {A, B2}, {A, B3}, {A, B4}, and {A, B5} makes this possible.

For conferencing scenarios, it is also useful to have a two-part Session-ID where the conference focus specifies one UUID. This might allow for correlation among the participants in a single conference, for example.

How a device acting on Session Identifiers stores, processes, or utilizes the Session Identifier is outside the scope of this document.

5. Transmitting the Session Identifier in SIP

Each session initiated or accepted MUST have a local UA-generated UUID associated with the session. This value MUST remain unchanged throughout the duration of that session.

A SIP UA MUST convey its Session Identifier UUID in all transmitted messages within the same session. To do this, each transmitted message MUST include the "Session-ID" header. The Session-ID header has the following ABNF [5] syntax:

```
  session-id    = "Session-ID" HCOLON local-uuid

                  *(SEMI sess-id-param)

  local-uuid    = sess-uuid

  remote-uuid   = sess-uuid

  sess-uuid     = 32(DIGIT / %x61-66)  ;32 chars of [0-9a-f]

  sess-id-param = remote-param / generic-param

  remote-param  = "remote" EQUAL remote-uuid
```

The productions "SEMI", "EQUAL", and "generic-param" production is defined in RFC 3261. The production DIGIT is defined in RFC 5234.

The Session-ID header MUST NOT have more than one "remote" parameter.

The "local-uuid" in the Session-ID header represents the UUID value of the UA transmitting the message. If the UA transmitting the message previously received a UUID value from its peer endpoint, it MUST include that UUID as the "remote" parameter. For example, using the UUID values from the previous section, a Session-ID header might appear like this:

```
  Session-ID: aeffa652b22911dfa81f12313a006823;
              remote=be11afc8b22911df86c412313a006823
```

The UUID values are presented as strings of lower-case hexadecimal characters, with the most significant byte of the UUID appearing first.

6. Endpoint Behavior

To comply with this specification, SIP UAs MUST include a Session-ID header-value in all messages transmitted as a part of a communication session.

A non-intermediary UAS that receives a Session-ID header MUST take note of the first UUID value that it receives in the Session-ID header and assume that that is the UUID of the peer endpoint within that communications session. UAs MUST include this received UUID

value as the "remote" parameter when transmitting subsequent messages.

It should be noted that messages received by a UA might contain a "remote" parameter that does match the UAs UUID. This might happen as a result of service interactions by intermediaries and MUST NOT be regarded as an error.

For any purpose the UA has for the Session-ID, it MUST assume that the Session-ID is {A,B} where "A" is the UUID value of this endpoint and "B" is the UUID value of the peer endpoint, taken from the most recently received message within this session.

An endpoint MUST assume that the UUID value of the peer UA MAY change at any time due to service interactions. However, once an UA allocates a UUID value for a communication session, the UA MUST NOT change that UUID value for the duration of the session, including when communication attempts are retried due to receipt of 4xx messages, when the session is redirected in response to a 3xx message, or when a session is transferred via a REFER message [6].

It is also important to note that if a session is forked by an intermediary in the network, the initiating UA may receive multiple responses back from different endpoints, each of which will contain a different UUID value. UAs MUST take care to ensure that the correct UUID value is returned in the "remote" parameter when responding to those endpoints.

7. Processing by Intermediaries

Intermediaries that wish to utilize the Session-ID MAY extract the UUID header-values from any SIP message. Alternatively, intermediaries MAY observe the first UUID value in the Session-ID header for messages sent in each direction and use those values to locally construct the Session Identifier.

Intermediaries MUST NOT alter the UUID values found in the Session-ID header, except as described in this section.

Intermediary devices that transfer a call, such as by joining together two different "call legs", MUST properly construct a Session-ID header that contains the correct UUID values and correct placement of those values. As described above, the recipient of any message initiated by the intermediary will assume that the first UUID value belongs to the peer endpoint.

If a SIP message having no Session-ID header is received by an intermediary, the intermediary MAY assign a "local-uuid" value to represent the sending endpoint and insert that value into all

signaling messages on behalf of the sending endpoint. If the intermediary is aware of a "remote" value that identifies the receiving UA, it MUST insert that value if also inserting the "local-uuid" value.

Devices that initiate communication sessions following the procedures for third party call control MUST fabricate a UUID value that will be utilized only temporarily. Once the responding endpoint provides a UUID value in a response message, the temporary value MUST be discarded and replaced with the endpoint-provided UUID value. Refer to the third-party call control example for an illustration.

Whenever there is a UA that does not implement this specification communicating through a B2BUA, the B2BUA MAY become dialog stateful and insert a UUID value into the Session-ID header on behalf of the UA according to the rules stated in Section 6.

8. Associating Endpoints in a Multipoint Conference

Multipoint Control Units (MCUs) group two or more sessions into a single multipoint conference. The MCU should utilize the same UUID value for each session that is grouped into the same conference. In so doing, each individual session in the conference will have a unique Session Identifier (since each endpoint will create a unique UUID of its own), but will also have one UUID in common with all other participants in the conference.

Intermediary devices, such as proxies or session border controllers, or network diagnostics equipment might assume that when they see two or more sessions with different Session Identifiers, but with one UUID in common, that the sessions are part of the same conference.

Note, however, that this assumption of being part of the same conference is not always true. For example, in a SIP forking scenario, there might also be what appears to be multiple sessions with a shared UUID value. This is actually desirable. What is desired is to allow for the association of related sessions. Whether sessions are related because of forking or because endpoints are communicating as a part of a conference does not matter. They are nonetheless related.

9. Various Call Flow Operations Utilizing the Session ID

Seeing something frequently makes understanding easier. With that in mind, we include several call flows with the initial UUID and the complete Session-ID indicated per message, as well as when the Session-ID changes according to the rules within this document during certain operations/functions.

9.1. Basic Session-ID Construction with 2 UUIDs

```
      Session-ID
         ---     Alice             B2BUA             Bob            Carol
         {A}       |----INVITE----->|                 |
         {A}       |                |----INVITE----->|
        {B,A}      |                |<---200 OK------|
        {B,A}      |<---200 OK------|                 |
        {A,B}      |------ACK------>|                 |
        {A,B}      |                |------ACK------>|
                   |<==============RTP==============>|
```

Figure 1 - Session-ID Creation when Alice calls Bob

Operation/Rules:

- o Transmitter of SIP message places its Session-ID UUID first in order;
- o UA-Alice sends its UUID in INVITE;
- o B2BUA receives an INVITE with a Session-ID header-value from UA-Alice, and transmits INVITE towards UA-Bob with an unchanged Session-ID header-value;
- o UA-Bob receives Session-ID and adds its UUID to construct the whole/complete Session-ID header-value in the 200 OK;
- o UA-Bob orders the UUIDs such that its UUID is first when UA-Bob is transmitting the SIP message;
- o B2BUA receives the 200 OK response with a complete Session-ID header-value from UA-Bob, and transmits 200 OK towards UA-Alice with an unchanged Session-ID header-value; while maintaining the order of UUIDs in the Session-ID header-value;
- o UA-Alice, upon reception of the 200 OK from the B2BUA, transmits the ACK towards the B2BUA with its UUID positioned first, and the UUID from UA-Bob positioned second in the Session-ID header-value.
- o B2BUA receives the ACK with a complete Session-ID header-value from UA-Alice, and transmits ACK towards UA-Bob with an unchanged Session-ID header-value; while maintaining the order of UUIDs in the Session-ID header-value;

9.2. Basic Call Transfer using REFER

From the example built within Section 9.1 (the basic session-ID establishment), we proceed to this 'Basic Call Transfer using REFER' example.

```
      Session-ID
         ---     Alice             B2BUA             Bob             Carol
                   |                 |                 |                |
                   |<==============RTP==============>|                |
        {B,A}      |                 |<---reINVITE----|                |
        {B,A}      |<---reINVITE----| (puts Alice on Hold)            |
        {A,B}      |-----200 OK---->|                 |                |
        {A,B}      |                 |-----200 OK---->|                |
        {B,A}      |                 |<-----ACK-------|                |
        {B,A}      |<-----ACK-------|                 |                |
                   |                 |                 |                |
        {B,A}      |                 |<----REFER------|                |
        {B,A}      |<----REFER------|                 |                |
        {A,B}      |-----200 OK---->|                 |                |
        {A,B}      |                 |-----200 OK---->|                |
        {B,A}      |                 |<-----ACK-------|                |
        {B,A}      |<-----ACK-------|                 |                |
        {A,B}      |-----NOTIFY---->|                 |                |
        {A,B}      |                 |-----NOTIFY---->|                |
        {B,A}      |                 |<----200 OK-----|                |
        {B,A}      |<----200 OK-----|                 |                |
                   |                 |                 |                |
         {A}       |-----INVITE---->|                                  |
         {A}       |                 |-----INVITE------------------->|
        {C,A}      |                 |<----200 OK--------------------|
        {C,A}      |<----200 OK-----|                                  |
        {A,C}      |------ACK------>|                                  |
        {A,C}      |                 |------ACK--------------------->|
                   |                 |                 |                |
                   |<=====================RTP======================>|
                   |                 |                 |                |
        {A,B}      |-----NOTIFY---->|                 |                |
        {A,B}      |                 |-----NOTIFY---->|                |
        {B,A}      |                 |<----200 OK-----|                |
        {B,A}      |<----200 OK-----|                 |                |
        {B,A}      |                 |<-----BYE-------|                |
        {B,A}      |<-----BYE-------|                 |                |
        {A,B}      |-----200 OK---->|                 |                |
        {A,B}      |                 |-----200 OK---->|                |
                   |                 |                 |                |
```

Figure 2 - Call Transfer using REFER

Operation/Rules:

Starting from the existing Alice/Bob call described in Figure 1, which established an existing Session-ID header-value...

- o UA-Bob reINVITEs Alice to call Carol, using a REFER transaction, as described in [RFC3515]. UA-Alice is initially put on hold, then told in the REFER who to contact with a new INVITE, in this case UA-Carol.

- o UA-Alice retains her UUID from the Alice-to-Bob call {A} when requesting a call with UA-Carol. This same UUID traverses the B2BUA unchanged.

- o UA-Carol receives the INVITE with a Session-ID UUID {A}, creates its own UUID {C}, and combines them to form a full Session-ID {C,A} in the 200 OK to the INVITE. This Session-ID header-value traverses the B2BUA unchanged towards UA-Alice.

- o UA-Alice receives the 200 OK with the Session-ID {C,A} and both responses to UA-Carol with an ACK, generates a NOTIFY to Bob with a Session-ID {A,B} indicating the call transfer was successful.

- o It does not matter which UA terminates the Alice-to-Bob call; Figure 2 shows UA-Bob doing this transaction.

9.3. Basic Call Transfer using reINVITE

From the example built within Section 9.1 (the basic session-ID establishment), we proceed to this 'Basic Call Transfer using reINVITE' example.

Alice is talking to Bob. Bob pushes a button on his phone to transfer Alice to Carol via the B2BUA (using reINVITE).

```
   Session-ID
      ---     Alice            B2BUA             Bob            Carol
                |                |                |               |
                |<==============RTP==============>|               |
                |                |                |               |
     {B,A}      |                |<---reINVITE----|               |
     {A,B}      |                |-----200 OK---->|               |
     {B,A}      |                |<-----ACK-------|               |
                |                |                |               |
      {A}       |                |-----INVITE-------------------->|
     {C,A}      |                |<----200 OK---------------------|
     {A,C}      |                |------ACK---------------------->|
                |                |                |               |
```

```
                   |<=====================RTP======================>|
                   |                |                |               |
        {B,A}      |                |<-----BYE-------|               |
        {B,A}      |<-----BYE-------|                |               |
        {A,B}      |-----200 OK---->|                |               |
        {A,B}      |                |-----200 OK---->|               |
                   |                |                |               |
```

Figure 3 - Call transfer using reINVITE

Operation/Rules:

- o We assume the call between Alice and Bob from Section 9.1 is operational with Session-ID {A,B}.
- o Bob sends a reINVITE to Alice to transfer her to Carol.
- o The B2BUA intercepts this reINVITE and sends a new INVITE with Alice's UUID {A} to Carol.
- o Carol receives the INVITE and accepts the request and adds her UUID {C} to the Session-ID for this session {C,A}.
- o Bob terminates the call (which Alice could too) with a BYE using their Session-ID {B,A}.

9.4. Single Focus Conferencing

Multiple users call into a conference server (say, an MCU) to attend one of many conferences hosted on or managed by that server. Each user has to identify which conference they want to join, but this information is not necessarily in the SIP messaging. It might be done by having a dedicated address for the conference or via an IVR, as assumed in this example. Each user in this example goes through a two-step process of signaling to gain entry onto their conference call.

```
      Session-ID                Conference
         ---     Alice            Focus             Bob            Carol
                   |                |                |               |
                   |                |                |               |
         {A}       |----INVITE----->|                |               |
        {M1,A}     |<---200 OK------|                |               |
        {A,M1}     |-----ACK------->|                |               |
                   |<====RTP=======>|                |               |
        {M',A}     |<---reINVITE----| (to change the |               |
        {A||M'}    |-----200 OK---->|   UUID to M')  |               |
        {M',A}     |<-----ACK-------|                |               |
                   |                |                |               |
```

```
                   |                 |                 |                |
         {B}       |                 |<----INVITE-----|                |
        {M2,B}     |                 |-----200 OK---->|                |
        {B,M2}     |                 |<-----ACK-------|                |
                   |                 |<=====RTP======>|                |
       {M'||B}     | (to change the  |----reINVITE--->|                |
       {B||M'}     |   UUID to M')   |<----200 OK-----|                |
       {M'||B}     |                 |------ACK------>|                |
                   |                 |                |                |
                   |                 |                |                |
         {C}       |                 |<--------------------INVITE-----|
        {M3,C}     |                 |---------------------200 OK---->|
        {C,M3}     |                 |<---------------------ACK-------|
                   |                 |<====================RTP======>|
       {M'||C}     | (to change the  |--------------------reINVITE--->|
       {C||M'}     |   UUID to M')   |<--------------------200 OK-----|
       {M'||C}     |                 |----------------------ACK------>|
```

Figure 4 - Single Focus Conference Bridge

Operation/Rules:

Alice calls into a conference server to attend a certain conference. This is a two-step operation since Alice cannot include the conference ID and any passcode in the INVITE.

o Alice sends an INVITE to the conference server with her UUID {A}.

o The conference server accepts using a generic, temporary UUID {M1}.

o Once Alice, the user, gains access to the IVR for this conference server, she enters a specific conference ID and whatever passcode (if needed) to enter a specific conference call.

o Once the conference server is satisfied Alice has identified which conference she wants to attend (including any passcode verification), the conference server reINVITEs Alice to the specific conference and includes the UUID {M'} for that conference. All valid participants in the same conference will receive this same UUID for identification purposes and to better enable monitoring, and tracking functions.

o Bob goes through this two-step process of an INVITE transaction, followed by a reINVITE transaction to get this same UUID for that conference.

   o In this example, Carol (and each additional user) goes through
     the same procedures and steps as Alice to get on this same
     conference.

9.5. Single Focus Conferencing using WebEx

Alice, Bob and Carol call into same Webex conference.

```
   Session-ID                 Conference
      ---     Alice             Focus             Bob             Carol
                |                 |                 |                |
                |<*** HTTPS ****>|                 |                |
                |  Transaction    |                 |                |
                |                 |                 |                |
      {M}       |<----INVITE-----|                 |                |
     {A||M}     |-----200 OK---->|                 |                |
     {M||A}     |<-----ACK-------|                 |                |
                |<=====RTP======>|                 |                |
                |                 |                 |                |
                |                 |<** HTTPS *****>|                |
                |                 |  Transaction    |                |
                |                 |                 |                |
      {M}       |                 |-----INVITE---->|                |
     {B||M}     |                 |<----200 OK-----|                |
     {M||B}     |                 |------ACK------>|                |
                |                 |<=====RTP======>|                |
                |                 |                 |                |
                |                 |<****************** HTTPS ***>|
                |                 |                    Transaction  |
                |                 |                 |                |
      {M}       |                 |--------------------INVITE----->|
     {C||M}     |                 |<-------------------200 OK------|
     {M||C}     |                 |---------------------ACK------->|
                |                 |<===================RTP=======>|
```

Figure 5 - Single Focus Webex Conference

Operation/Rules:

   o Alice communicates with Webex server with desire to join a
     certain meeting, by meeting number; also includes UA-Alice's
     contact information (phone number or URI).

   o Conference Focus server sends INVITE to UA-Alice to start
     session with the Session-ID of that server for this A/V
     conference call.

   o Bob and Carol perform same function to join this same A/V
     conference call as Alice.

9.6. Cascading Conference Bridge Support for the Session-ID

To expand conferencing capabilities requires cascading conference bridges. A conference bridge, or MCU, needs a way to identify itself when contacting another MCU. RFC 4579 [6] defines the 'isfocus' Contact: header parameter just for this purpose.

Cascading MCUs for the purpose of having each use the same UUID (aka half the Session-ID), in its simplest form, is one MCU informing another which UUID to use for joining UAs.

```
      Session-ID
         ---     MCU-1            MCU-2            MCU-3           MCU-4
                   |                |                |               |
         {M'}      |----INVITE----->|                |               |
         {M'}      |<---200 OK------|                |               |
         {M'}      |-----ACK------->|                |               |
```

Figure 6 - MCUs Communicating Session-ID UUID for Bridge

Regardless of which MCU (1 or 2) a UA contacts for this conference, once the above exchange has been received and acknowledged, the UA will get the same M' UUID from the MCU for the complete Session-ID.

A more complex form would be a series of MCUs all being informed of the same UUID to use for a specific conference. This series of MCUs can either be informed

- o All by one MCU (that initially generates the UUID for the conference),
- o The one MCU that generates the UUID informs one or several MCUs of this common UUID, and they inform downstream MCUs of this common UUID each will be using for this one conference, or

```
      Session-ID
         ---     MCU-1            MCU-2            MCU-3           MCU-4
                   |                |                |               |
         {M'}      |----INVITE----->|                |               |
         {M'}      |<---200 OK------|                |               |
         {M'}      |-----ACK------->|                |               |
                   |                |                |               |
         {M'}      |---------------------INVITE----->|               |
         {M'}      |<--------------------200 OK------|               |
         {M'}      |----------------------ACK------->|               |
                   |                |                |               |
         {M'}      |-------------------------------------INVITE----->|
         {M'}      |<------------------------------------200 OK------|
         {M'}      |--------------------------------------ACK------->|
```

Figure 7 - MCU Communicating Session-ID UUID to More than One

Operation/Rules:

- o The MCU generating the Session-ID UUID communicates this in a separate INVITE, having a Contact header with the 'isfocus' header parameter. This will identify the MCU as what RFC 4579 conference-aware SIP entity.

- o The MCU that is contacted, i.e., the UAS MCU, does not populate or complete the Session-ID header value. The UAS MCU transmits a 200 OK response acknowledging it is to respond with this M' UUID to all requests for the designated conference.

- o An MCU that receives this M' UUID in an inter-MCU transaction, can communicate the M' UUID in a manner in which it was received (though this time this second MCU would be the UAC MCU), unless local policy dictates otherwise.

9.7. Basic 3PCC for two UAs

External entity sets up call to both Alice and Bob for them to talk to each other.

```
   Session-ID
      ---     Alice            B2BUA             Bob            Carol
                |                |                |
      {X}       |<----INVITE-----|                |
     {A,X}      |-----200 OK---->|                |
      {A}       |                |----INVITE----->|
     {B,A}      |                |<---200 OK------|
     {A,B}      |<-----ACK-------|                |
     {A,B}      |                |------ACK------>|
                |<==============RTP==============>|
```

Figure 8 - 3PCC initiated call between Alice and Bob

Operation/Rules:

- o Some out of band procedure directs a B2BUA (or other SIP server) to have Alice and Bob talk to each other.

- o The SIP server INVITEs Alice to a session and uses a temporary UUID {X}.

- o Alice receives and accepts this call set-up and includes her UUID {A} in the Session-ID, now {A,X}.

- o The SIP server uses Alice's UUID {A}, and discards its own {X} to INVITE Bob to the session as if this came from Alice originally.

- o Bob receives and accepts this INVITE and adds his own UUID {B} to the Session-ID, now {B,A} for the response.

- o And the session is established.

10. Compatibility with a Previous Implementation

There is a much earlier and proprietary document that specifies the use of a Session-ID header that we will herewith attempt to achieve backwards compatibility. Neither Session-ID has any versioning information, so merely adding that this document describes "version 2" is insufficient. Here are the set of rules for compatibility between the two specifications. For the purposes of this discussion, we will label the proprietary specification of the Session-ID as the "old" version and this specification as the "new" version of the Session-ID.

The previous (i.e., "old") version only has a single value as a Session-ID, but has a generic-parameter value that can be of use.

In order to have an "old" version talk to an "old" version implementation, nothing needs to be done as far as the IETF is concerned.

In order to have a "new" version talk to a "new" version implementation, both implementations need to following this document (to the letter) and everything should be just fine.

In order to have an "old" version talk to a "new" version implementation, several aspects need to be looked at. They are:

- o The "old" version UA will include a single UUID as its Session-ID.

- o The "new" version UA will respond by including a complete Session-ID with two UUIDs, with the "new" version's UUID listed first (because it cannot know it is talking with an "old" version implementation at this point).

- o The "old" version UA will have to ignore the first UUID, and consider its singular "old" UUID as valid, as long as the value does not change..

- o During subsequent transactions within this session, the "new" version may receive SIP requests without its UUID, but with the

"old" version's UUID. The "new" version UA MUST add its UUID to the received Session-ID. The "old" version implementation will merely disregard it each time it receives this "new" version UUID (if it was not the first UUID).

In order to have a "new" version talk to an "old" Version implementation, several aspects need to be looked at. They are:

- The "new" version UA will include a single UUID as its initial Session-ID header always, not knowing which version of UA it is communicating with.

- The "old" version UA will respond by seeing the UUID as a valid and complete Session-ID and not include another UUID or generic-param. Thus, the 200 OK will not include any Session-ID part of its own from the "old" version implementation.

Rule: implementation supporting a "new" version of the Session-ID MUST NOT error or otherwise reject receiving only its own UUID back in any transaction. It MUST interpret this response to mean that it is communicating with an "old" Session-ID implementation.

- Open question - how do we want all intermediaries and/or monitoring systems to interpret this single UUID complete Session-ID?

11. Security Considerations

When creating a UUID value, endpoints SHOULD ensure that there is no user or device-identifying information contained within the UUID. In some environments, though, use of a MAC address, which is one option when constructing a UUID, may be desirable, especially in some enterprise environments. When communicating over the Internet, though, the UUID value MUST utilize random values.

The Session-ID might be utilized for logging or troubleshooting, but MUST NOT be used for billing purposes. { Why does this matter? }

Other considerations???

12. IANA Considerations

The following is the registration for the 'Session-ID' header field to the "Header Name" registry at http://www.iana.org/assignments/sip-parameters:

RFC number: [this document]

Header name: 'Session-ID'

Compact form: none

13. Acknowledgments

The authors would like to than Hadriel Kaplan, Christer Holmberg, and Paul Kyzivat for their invaluable comments during the development of this document.

This document was prepared using 2-Word-v2.0.template.dot.

14. References

14.1. Normative References

[1] Rosenberg, J., et al., "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Recommendation ITU-T H.323, "Packet-based multimedia communications systems", December 2009.

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

[4] Leach, P., Mealling, M., Salz, R., "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

[5] Crocker, D., Overell, P, "Augmented BNF for Syntax Specifications: ABNF", RFC 5234, January 2008.

[6] A. Johnston, O. Levin, "Session Initiation Protocol (SIP) Call Control - Conferencing for User Agents", RFC 4579, August 2006

Informative References

[6] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[7] Schulzrinne, H., et al., "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003.

[8] Jones, et al., "Requirements for an End-to-End Session Identification in IP-Based Multimedia Communication Networks", draft-jones-insipid-session-id-reqts-02.txt, October 2012.

Author's Addresses

Paul E. Jones
Cisco Systems, Inc.
7025 Kit Creek Rd.
Research Triangle Park, NC 27709
USA

Phone: +1 919 476 2048
Email: paulej@packetizer.com
IM: xmpp:paulej@packetizer.com

Chris Pearce
Cisco Systems, Inc.
2300 East President George Bush Highway
Richardson, TX 75082
USA

Phone: +1 972 813 5123
Email: chrep@cisco.com
IM: xmpp:chrep@cisco.com

James Polk
Cisco Systems, Inc.
3913 Treemont Circle
Colleyville, Texas
USA

Phone: +1 817 271 3552
Email: jmpolk@cisco.com
IM: xmpp:jmpolk@cisco.com

Gonzalo Salgueiro
Cisco Systems, Inc.
7025 Kit Creek Rd.
Research Triangle Park, NC 27709
USA

Phone: +1 919 392 3266
Email: gsalguei@cisco.com
IM: xmpp:gsalguei@cisco.com