Network Working Group
Internet Draft
Intended status: Standards Track
Expires: April 22, 2013

P. Jones
C. Pearce
J. Polk
G. Salgueiro
Cisco Systems
October 22, 2012

End-to-End Session Identification in IP-Based Multimedia Communication Networks
draft-jones-insipid-session-id-01.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.

Table of Contents

1. Introduction....2
2. Conventions used in this document....3
3. Session Identifier Requirements and Use Cases....3
4. Constructing the Session Identifier....3
5. Transmitting the Session Identifier in SIP....4
6. Endpoint Behavior....5
7. Processing by Intermediaries....6
8. Associating Endpoints in a Multipoint Conference....6
9. Correlating Media Flows with Sessions....7
10. Various Call Flow Operations Utilizing the Session ID....7
10.1. Basic Session-ID Construction with 2 UUIDs....7
10.2. Basic Call Transfer using REFER....8
10.3. Basic Call Transfer using reINVITE....10
10.4. Single Focus Conferencing....11
10.5. Single Focus Conferencing using WebEx....12
10.6. Basic 3PCC for two UAs....13
11. Compatibility with a Previous Implementation....14
12. Security Considerations....15
13. IANA Considerations....15
14. Acknowledgments....16
15. References....16
15.1. Normative References....16
15.2. Informative References....16
Author's Addresses....17

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, billing, 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 [5]. 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" [5].

4. Constructing the Session Identifier

The Session Identifier is comprised of two RFC 4122 defined UUIDs [4], with each UUID created by 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.

Devices that act upon the Session Identifier may wish to represent these UUID values in some manner other than a pair of distinct values for, as examples, logging or internal value comparison. A device MAY take the two UUID values and produce a single 32-octet binary value that can be efficiently compared with other values. When constructing a single binary value out of the component UUIDs, it is RECOMMENDED that devices perform a binary comparison of the two UUIDs, starting with the most significant byte of each UUID. The UUID with the higher binary value is placed after the UUID with the lower binary value. As an example, if the Session Identifier {A,B} is stored and treated as a single binary value and "A" is numerically greater than "B", then the two values would be concatenated as B||A. When only one UUID value is known, entities MAY assume the absent UUID has a value of zero (i.e., 16 octets with a zero value).

Consider the following example.

Endpoint 1 produces this UUID: 0xaeffa652b22911dfa81f12313a006823

Endpoint 2 produces this UUID: 0xbe11afc8b22911df86c412313a006823

The resulting concatenated Session Identifier would be:
0xaeffa652b22911dfa81f12313a006823be11afc8b22911df86c412313a006823

In the above example, the UUIDs are presented as a string of hexadecimal characters that correspond to the binary values comprising the UUID as shown in the table at the end of Section 4.1.2 of RFC 4122 [4].

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 syntax:

```
Session-ID = "Session-ID" HCOLON sess-id
             ( SEMI rcvr-uuid )
             *( SEMI generic-param )

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

rcvr-uuid  =  "rcvr" EQUAL 32(DIGIT / %x61-66)
```

The "sess-id" value 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 "rcvr" parameter. For example, using the UUID values from the previous section, a Session-ID header might appear like this:

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

The UUID values are presented as strings of 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. Session-ID header-values MUST NOT be present in any other SIP header than the Session-ID header.

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 "rcvr" parameter when transmitting subsequent messages.

UAs MUST ignore the value in the "rcvr" parameter in any message it receives, as this value may be incorrect due to service interactions as shown in examples later in this document.

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 "rcvr" 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.

If performing interworking between SIP and another session protocol, an intermediary MUST convert the Session-ID header as necessary so that it properly places the correct UUID value of the message transmitter and assumed recipient. This is a protocol gateway or interworking function.

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.

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.

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. Correlating Media Flows with Sessions

It may be desirable to insert the Session Identifier header-value into media-related packets, such as RSVP messages or RTCP packets. In so doing, it is possible for network elements to

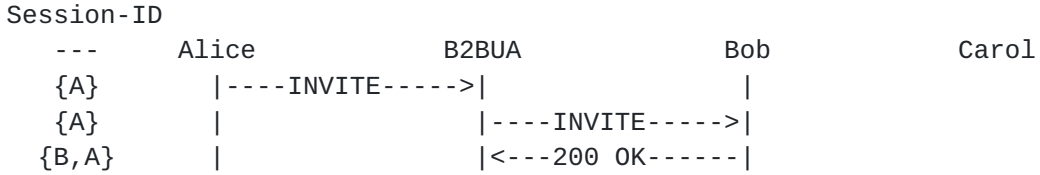
1. correlate session signaling with media flows;
2. associate multiple media flows with a single session; and
3. associate multiple media flows from multiple devices that are part of a single conference

Notwithstanding the foregoing, the use of the Session Identifier for purposes other than end-to-end session identification is outside the scope of this document.

10. 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.

10.1. Basic Session-ID Construction with 2 UUIDs



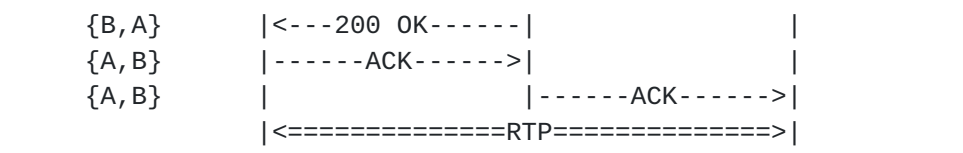
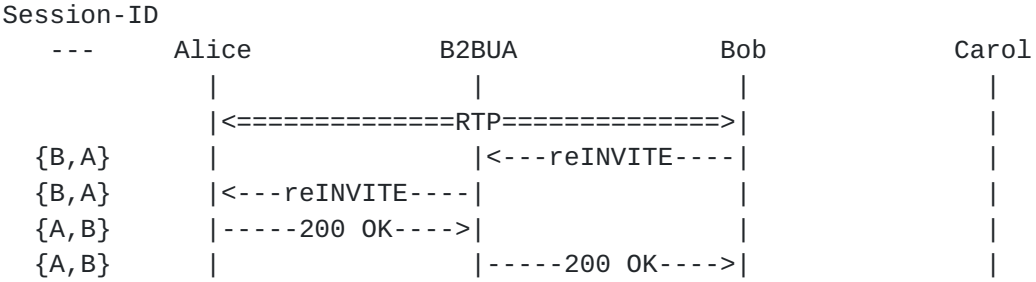


Figure 1 - Session-ID Creation when Alice calls Bob

Operation/Rules:

- Transmitter of SIP message places its Session-ID UUID first in order;
- UA-Alice sends its UUID in INVITE;
- 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;
- UA-Bob receives Session-ID and adds its UUID to construct the whole/complete Session-ID header-value in the 200 OK;
- UA-Bob orders the UUIDs such that its UUID is first when UA-Bob is transmitting the SIP message;
- 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;
- 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.
- 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;

10.2. Basic Call Transfer using REFER



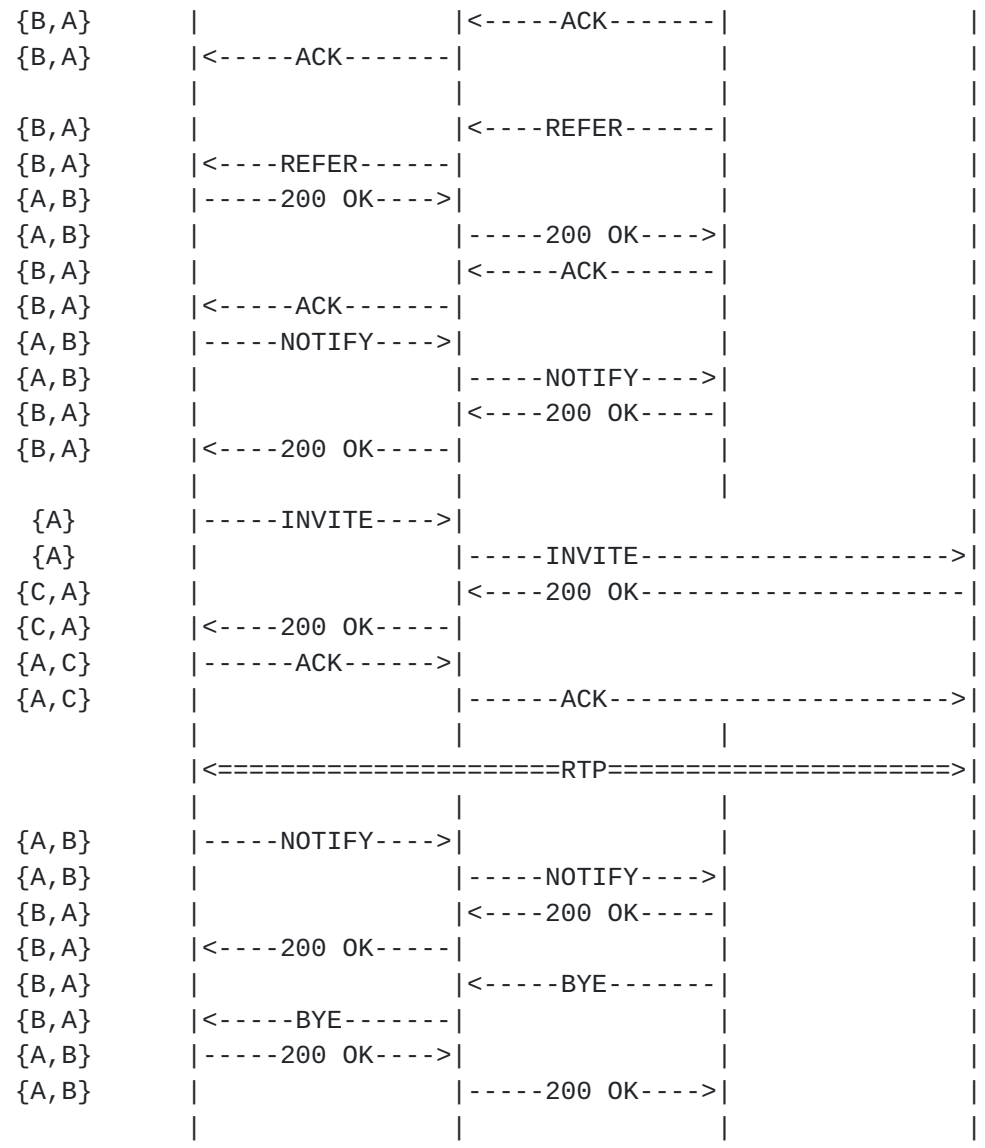


Figure 2 - Call Transfer using REFER

Operation/Rules:

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

- 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.

- 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.

10.3. Basic Call Transfer using reINVITE

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

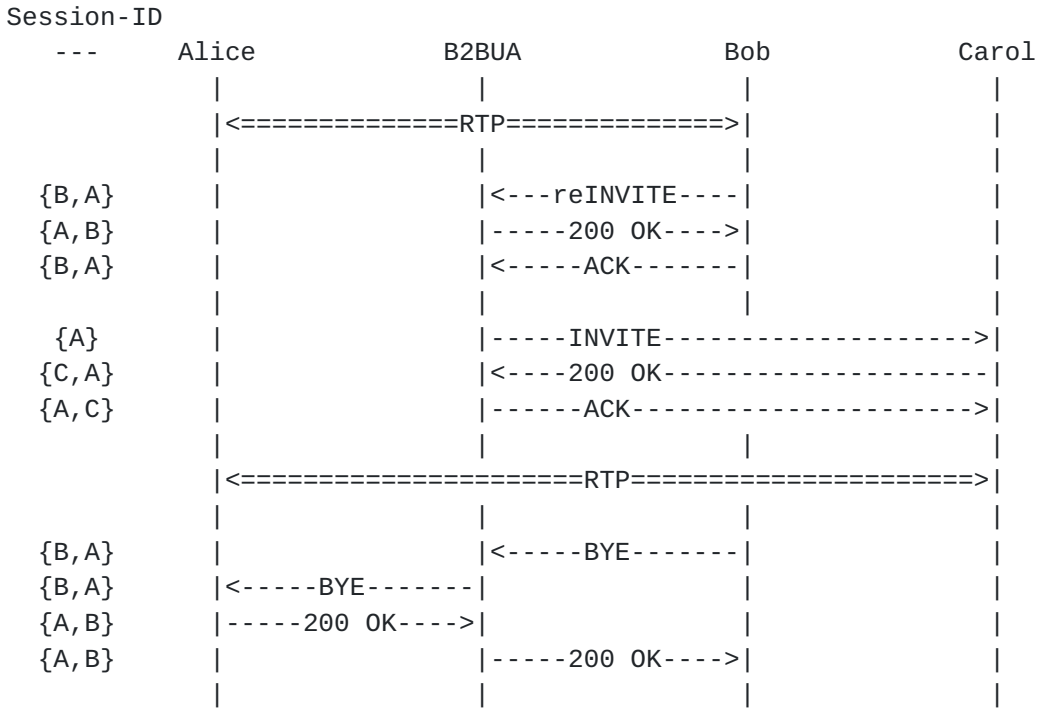


Figure 3 - Call transfer using reINVITE

Operation/Rules:

- o We assume the call between Alice and Bob from Section 10.1 is operational with Session-ID {A,B}.

- o Bob sends a reINVITE to Alice to transfer her to Carol.

- The B2BUA intercepts this reINVITE and sends a new INVITE with Alice's UUID {A} to Carol.

- Carol receives the INVITE and accepts the request and adds her UUID {C} to the Session-ID for this session {C,A}.

- Bob terminates the call (which Alice could too) with a BYE using their Session-ID {B,A}.

10.4. 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 in the SIP messaging. Rather, it is done via an IVR. Thus, each user goes through a two-step process of signaling to gain entry onto their conference call.

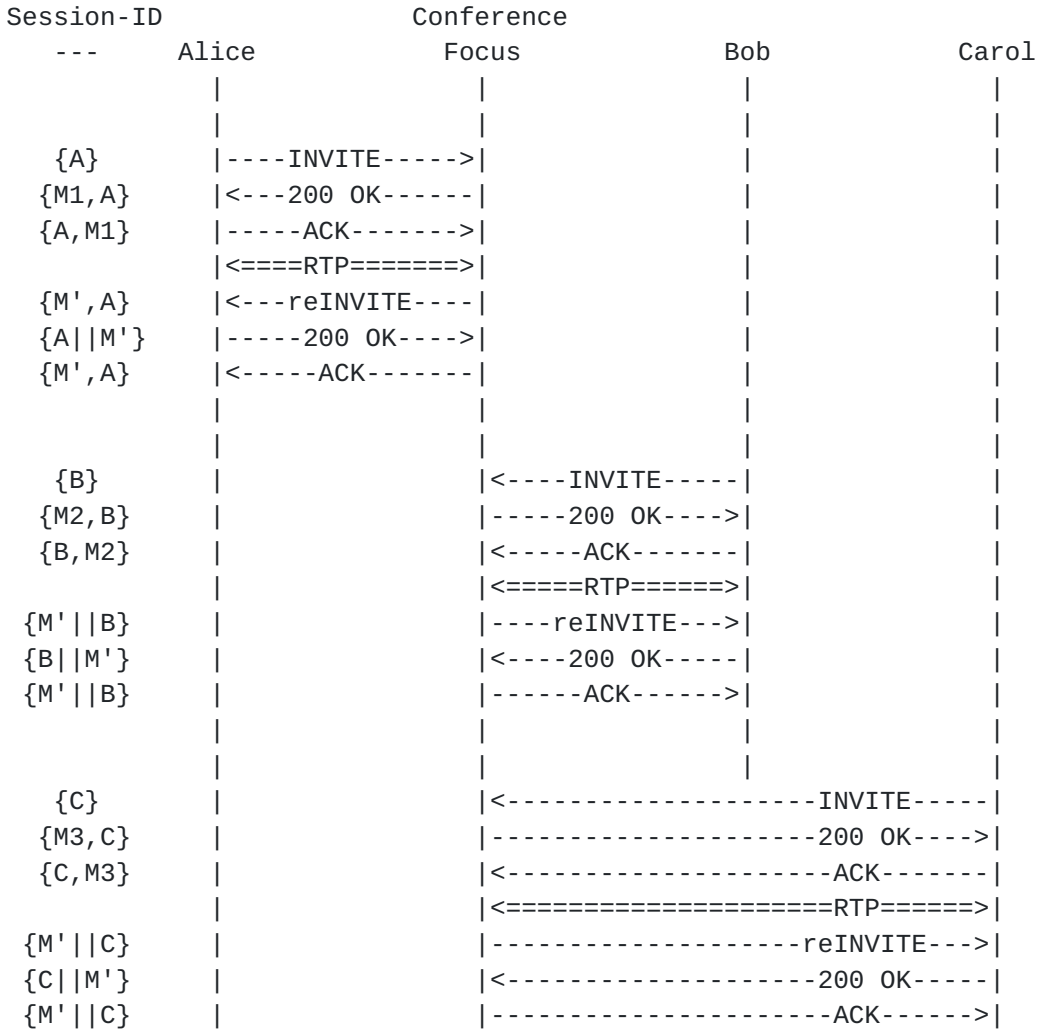


Figure 4 - Single Focus Conference Bridge

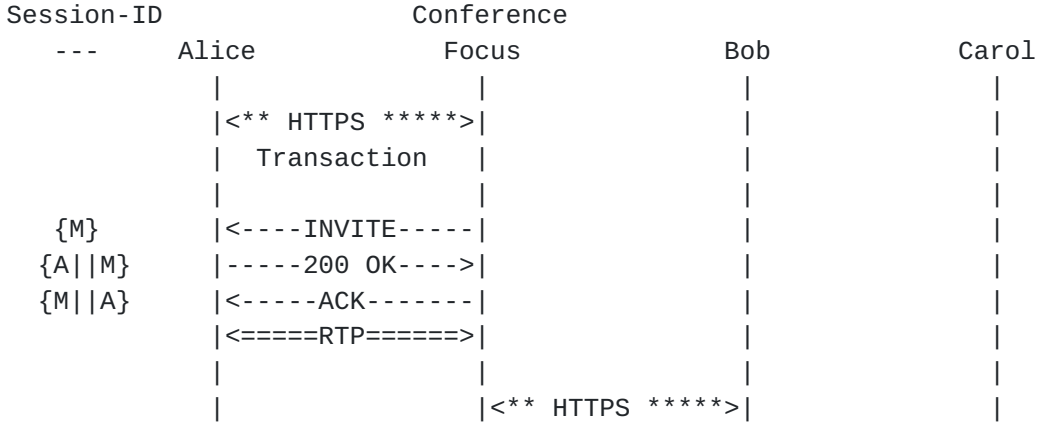
Operation/Rules:

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.

- Alice sends an INVITE to the conference server with her UUID {A}.
- The conference server accepts using a generic, temporary UUID {M1}.
- 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.
- 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, tracking and billing functions.
- Bob goes through this two-step process of an INVITE transaction, followed by a reINVITE transaction to get this same UUID for that conference.
- In this example, Carol (and each additional user) goes through the same procedures and steps as Alice to get on this same conference.

10.5. Single Focus Conferencing using WebEx

Alice, Bob and Carol call into same Webex conference.



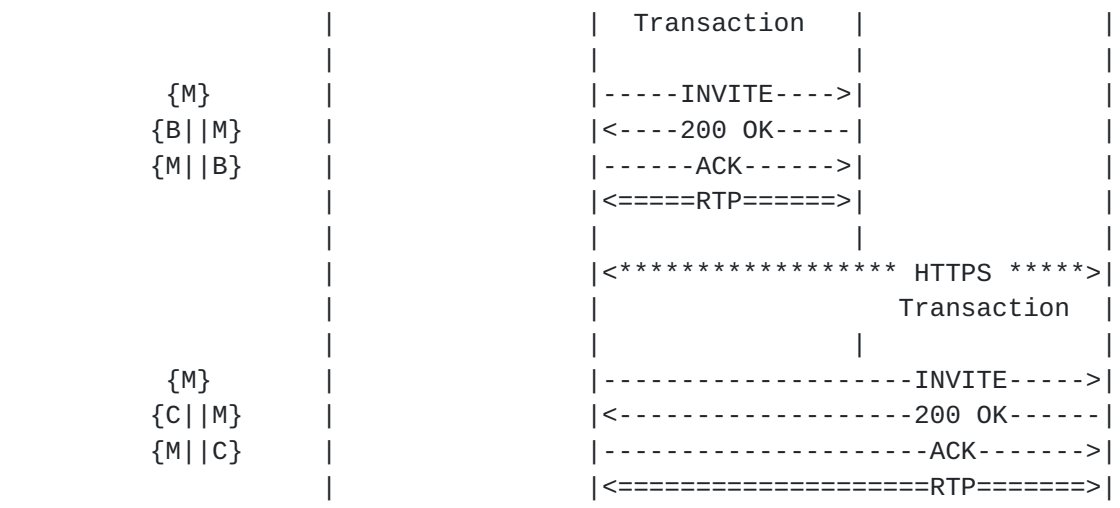


Figure 5 - Single Focus Webex Conference

Operation/Rules:

- 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).

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

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

10.6. Basic 3PCC for two UAs

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

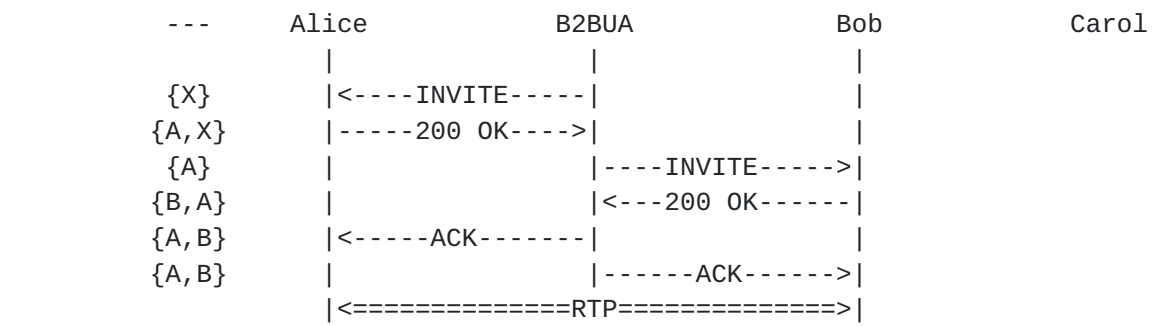


Figure 6 - 3PCC initiated call between Alice and Bob

Operation/Rules:

- Some out of band procedure directs a B2BUA (or other SIP server) to have Alice and Bob talk to each other.
- The SIP server INVITEs Alice to a session and uses a temporary UUID {X}.
- Alice receives and accepts this call set-up and includes her UUID {A} in the Session-ID, now {A,X}.
- 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.
- Bob receives and accepts this INVITE and adds his own UUID {B} to the Session-ID, now {B,A} for the response.
- And the session is established.

11. 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 version 0 and this specification as version 1 of the Session-ID.

The previous 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 a Version 0 talk to a Version 0 implementation, nothing needs to be done as far as the IETF is concerned.

In order to have a Version 1 talk to a Version 1 implementation, both implementations need to following this document (to the letter) and everything should be just fine.

In order to have a Version 0 talk to a Version 1 implementation, several aspects need to be looked at. They are:

- The Version 0 UA will include a single UUID as its Session-ID.
- The Version 1 UA will respond by including a complete Session-ID with two UUIDs, with the Version 1's UUID listed first (because it cannot know it is talking with a Version 0 implementation at this point).

- The Version 0 UA will have to ignore the first UUID and react to the second UUID, which would be the recipient's Session-ID.

- During subsequent transactions within this session, the Version 1 may receive SIP requests without its UUID, but with the Version 0's UUID. The Version 1 UA MUST add its UUID to the received Session-ID. The Version 0 implementation will merely disregard it each time it receives this Version 1 UUID (if it was not the first UUID).

In order to have a Version 1 talk to a Version 0 implementation, several aspects need to be looked at. They are:

- The Version 1 UA will include a single UUID as its initial Session-ID header always, not knowing which version of UA it is communicating with.

- The Version 0 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 Version 0 implementation.

- Open question - How do we want the Version 1 implementation interpret this?

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

12. 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.

Other considerations???

13. 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

14. Acknowledgments

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

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

15. References

15.1. 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] 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.

15.2. Informative References

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

[7] Braden, R., et al., "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

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

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, USAUSA

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