Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2015

P. Kyzivat
L. Xiao
C. Groves
Huawei
R. Hansen
Cisco Systems
July 4, 2014

CLUE Signaling
draft-ietf-clue-signaling-02

Abstract

This document specifies how CLUE-specific signaling such as the CLUE protocol [I-D.presta-clue-protocol] and the CLUE data channel [I-D.ietf-clue-datachannel] are used with each other and with existing signaling mechanisms such as SIP and SDP to produce a telepresence call.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on January 5, 2015.

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

To enable devices to participate in a telepresence call, selecting the sources they wish to view, receiving those media sources and displaying them in an optimal fashion, CLUE involves two principal and inter-related protocol negotiations. SDP, conveyed via SIP, is used to negotiate the specific media capabilities that can be delivered to specific addresses on a device. Meanwhile, a CLUE protocol [I-D.presta-clue-protocol], transported via a CLUE data channel [I-D.ietf-clue-datachannel], is used to negotiate the capture sources available, their attributes and any constraints in their use, along which which captures the far end provides a device wishes to receive.

Beyond negotiating the CLUE channel, SDP is also used to negotiate the details of supported media streams and the maximum capability of each of those streams. As the CLUE Framework [I-D.ietf-clue-framework] defines a manner in which the media provider expresses their maximum encoding capabilities, SDP is also used to express the encoding limits for each potential encoding.

Backwards-compatibility is an important consideration of the document: it is vital that a CLUE-capable device contacting a device that does not support CLUE is able to fall back to a fully functional non-CLUE call. The document also defines how a non-CLUE call may be upgraded to CLUE in mid-call, and similarly how CLUE functionality can be removed mid-call to return to a standard non-CLUE call.

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 [RFC2119].

This document draws liberally from the terminology defined in the CLUE Framework [I-D.ietf-clue-framework].

Other terms introduced here:

CLUE data channel: A reliable, bidirectional, transport mechanism used to convey CLUE messages. See [I-D.ietf-clue-datachannel] for more details.

CLUE-capable device: A device that supports the CLUE data channel [I-D.ietf-clue-datachannel], the CLUE protocol [I-D.presta-clue-protocol] and the principles of CLUE negotiation, and wishes to upgrade the call to CLUE-enabled status.

CLUE-enabled call: A call in which two CLUE-capable devices have successfully negotiated support for a CLUE data channel in SDP. A CLUE-enabled call is not necessarily immediately able to send CLUE-controlled media; negotiation of the data channel and of the CLUE protocol must complete first. Calls between two CLUE-capable devices which have not yet successfully completed negotiation of support for the CLUE data channel in SDP are not considered CLUE-enabled.

Non-CLUE device: A device that supports standard SIP and SDP, but either does not support CLUE, or that does but does not currently wish to invoke CLUE capabilities.

CLUE-controlled media: A media "m" line that is under CLUE control; the capture source that provides the media on this "m" line is negotiated in CLUE. See Section 4 for details of how this control is signalled in SDP. There is a corresponding "non-CLUE-controlled" media term.

3. Media Feature Tag Definition

The "sip.clue" media feature tag indicates support for CLUE. A CLUE-capable device SHOULD include this media feature tag in its REGISTER requests and OPTION responses. It SHOULD also include the media feature tag in INVITE and UPDATE [RFC3311] requests and responses.

Presence of the media feature tag in the contact field of a request or response can be used to determine that the far end supports CLUE.

4. SDP Grouping Framework CLUE Extension Semantics

4.1. General

This section defines a new SDP Grouping Framework extension, CLUE.

The CLUE extension can be indicated using an SDP session-level 'group' attribute. Each SDP media "m" line that is included in this group, using SDP media-level mid attributes, is CLUE-controlled, by a CLUE data channel also included in this CLUE group.

Currently only support for a single CLUE group is specified. A device MUST NOT include more than one CLUE group in its SDP unless it is following a specification that defines how multiple CLUE channels are defined, and is either able to determine that the other side of the SDP exchange supports multiple CLUE channels, or is able to fail gracefully in the event it does not.

4.2. The CLUE data channel and the CLUE grouping semantic

The CLUE data channel [I-D.ietf-clue-datachannel] is a bidirectional SCTP over DTLS channel used for the transport of CLUE messages. This channel must be established before CLUE protocol messages can be exchanged and CLUE-controlled media can be sent.

The data channel is a generic transport that is not specific to CLUE - if a device wishes to use the CLUE protocol on the data channel it MUST include a CLUE group in the SDP and include the "mid" of the "m" line for the data channel in that group. A CLUE group MUST include the "mid" of the "m" line for one (and only one) data channel, and the "mid" of the "m" line of a data channel "mid" MUST NOT be included in more than one CLUE group.

Presence of the data channel in a CLUE group in an SDP offer or answer also serves, along with the "sip.clue" media feature tag, as an indication that the device supports CLUE and wishes to upgrade the call to include CLUE-controlled media. A CLUE-capable device SHOULD include a data channel "m" line in offers and, when allowed by [RFC3264], answers.

4.3. CLUE-controlled media and the CLUE grouping semantic

CLUE-controlled media lines in an SDP are "m" lines in which the content of the media streams to be sent is negotiated via the CLUE protocol [I-D.presta-clue-protocol]. For an "m" line to be CLUE-controlled, its "mid" value MUST be included in a CLUE group. CLUE-controlled media line "mid"s MUST NOT be included in more than one CLUE group.

CLUE-controlled media is controlled by the CLUE protocol as negotiated on the CLUE data channel with an "mid" included in the CLUE group. If negotiation of the data channel in SDP failed due to lack of CLUE support by the remote device or for any other reason the other "m" lines in the group are still considered CLUE-controlled and under all the restrictions of CLUE-controlled media specified in this document.

"m" lines not specified as under CLUE control follow normal rules for media streams negotiated in SDP as defined in documents such as [RFC3264].

4.4. SDP semantics for CLUE-controlled media

4.4.1. Signalling CLUE Encodings

The CLUE Framework [I-D.ietf-clue-framework] defines the concept of "encodings", which represent the sender's encode ability. Each encoding the media provider wishes to signal is signalled via an "m" line of the appropriate media type, which MUST be marked as sendonly with the "a=sendonly" attribute or as inactive with the "a=inactive" attribute.

The encoder limits of active (eg, "a=sendonly") encodings can then be expressed using existing SDP syntax. For instance, for H.264 see Table 6 in [RFC6184] for a list of valid parameters for representing encoder sender stream limits.

These encodings are CLUE-controlled and hence MUST include an "mid" in a CLUE group as defined above.

As well as the normal restrictions defined in [RFC3264] the stream MUST be treated as if the "m" line direction attribute had been set to "a=inactive" until the media provider has received a valid CLUE CONFIGURE message specifying the capture to be used for this stream. This means that media packets MUST NOT be sent until configuration is complete, while non-media packets such as STUN and DTLS MUST be sent as normal if negotiated.

Every "m" line representing a CLUE encoding MUST contain a "label" attribute as defined in [RFC4574]. This label is used to identify the encoding by the sender in CLUE ADVERTISEMENT messages and by the receiver in CLUE CONFIGURE messages. Each label used for a CLUE-controlled "m" line MUST be different from the label on all other "m" lines in the same CLUE group in the SDP message.

4.4.1.1. Referencing encodings in the CLUE protocol

CLUE encodings are defined in SDP, but can be referenced from CLUE protocol messages - this is how the protocol defines which encodings are part of an encoding group (in ADVERTISEMENT messages) and which encoding with which to encode a specific capture (in CONFIGURE messages). The labels on the CLUE-controlled "m" lines are the references that are used in the CLUE protocol.

Each <encID> element in a CLUE ADVERTISEMENT message SHOULD represents an encoding defined in SDP; the specific encoding referenced is a CLUE-controlled "m" line in the most recent SDP sent by the sender of the ADVERTISMENT message with a label value corresponding to the text content of the <encID>.

Similarly, each <encID> element in a CLUE CONFIGURE message SHOULD

represents an encoding defined in SDP; the specific encoding referenced is a CLUE-controlled "m" line in the most recent SDP received by the sender of the CONFIGURE message with a label value corresponding to the text content of the <encID>.

Note that the non-atomic nature of SDP/CLUE protocol interaction may mean that there are temporary periods where an <encID> in a CLUE message does not reference an SDP "m" line, or where an encoding represented in SDP is not referenced in a CLUE protocol message. See Section 5 for specifics.

4.4.1.2. Media line directionality

Presently, this specification mandates that CLUE-controlled "m"-lines must be unidirectional. This is because setting "m"-lines to "a=sendonly" allows the encoder limits to be expressed, whereas in other cases codec attributes express the receive capabilities of a media line.

It is possible that in future versions of this draft or its successor this restriction will be relaxed. If a device does not feel there is a benefit to expressing encode limitations, or if there are no meaningful codec-specific limitations to express (such as with many audio codecs) there are benefits to allowing bidirectional "m"-lines. With bidirectional media lines recipients do not always need to create a new offer to add their own "m"-lines to express their send capabilities; if they can produce an equal or lesser number of streams to send then they may not need additional "m"-lines.

However, at present the need to express encode limitations and the wish to simplify the offer/answer procedure means that for the time being only unidirectional media lines are allowed for CLUE-controlled media. The highly asymmetric nature of CLUE means that the probability of the recipient of the initial offer needing to make their own offer to add additional "m"-lines is significantly higher than it is for most other SIP call scenarios, in which there is a tendancy for both sides to have similar numbers of potential audio and video streams they can send.

4.4.2. Negotiating receipt of CLUE capture encodings in SDP

A receiver who wishes to receive a CLUE stream via a specific encoding requires an "a=recvonly" "m" line that matches the "a=sendonly" encoding.

These "m" lines are CLUE-controlled and hence MUST include their "mid" in the CLUE group corresponding to the CLUE group of encoding they wish to receive.

4.5. SDP Offer/Answer Procedures

4.5.1. Generating the Initial Offer

A CLUE-capable device sending an initial SDP offer of a SIP session SHOULD include an "m" line for the data channel to convey the CLUE protocol, along with a CLUE group containing the "mid" of the data channel "m" line.

For interoperability with non-CLUE devices a CLUE-capable device sending an initial SDP offer SHOULD NOT include any "m" line for CLUE-controlled media beyond the "m" line for the CLUE data channel, and SHOULD include at least one non-CLUE-controlled media "m" line.

If the device has evidence that the receiver is also CLUE-capable, for instance due to receiving an initial INVITE with no SDP but including a "sip.clue" media feature tag, the above recommendation is waived, and the initial offer MAY contain "m" lines for CLUE-controlled media.

With the same interoperability recommendations as for encodings, the sender of the initial SDP offer MAY also include "a=recvonly" media lines to preallocate "m" lines to receive media. Alternatively, it MAY wait until CLUE protocol negotiation has completed before including these lines in a new offer/answer exchange - see Section 5 for recommendations.

4.5.2. Generating the Answer

4.5.2.1. Negotiating use of CLUE and the CLUE data channel

If the recipient is CLUE-capable and the initial offer contains both an "m" line for a data channel and a CLUE group containing the "mid" for that "m" line, they SHOULD negotiate data channel support for an "m" line, and include the "mid" of that "m" line in a corresponding CLUE group.

A CLUE-capable recipient that receives an "m" line for a data channel but no corresponding CLUE group containing the "mid" of that "m" line SHOULD include a corresponding data channel "m" line if there are any other non-CLUE protocols it can convey over that channel, otherwise it SHOULD NOT negotiate the data channel.

4.5.2.2. Negotiating CLUE-controlled media

If the initial offer contained "a=recvonly" CLUE-controlled media lines the recipient SHOULD include corresponding "a=sendonly" CLUE-controlled media lines, up to the maximum number of encodings it

wishes to advertise. As CLUE-controlled media, the "mid" of these "m" lines must be included in the corresponding CLUE group.

If the initial offer contained "a=sendonly" CLUE-controlled media lines the recipient MAY include corresponding "a=recvonly" CLUE-controlled media lines, up to the maximum number of capture encodings it wishes to receive. Alternatively, it MAY wait until CLUE protocol negotiation has completed before including these lines in a new offer/answer exchange - see Section 5 for recommendations.

4.5.2.3. Negotating non-CLUE controlled media

A CLUE-controlled device implementation MAY prefer to render initial, single-stream audio and/or video for the user as rapidly as possible, transitioning to CLUE-controlled media once that has been negotiated. Alternatively, an implementation may wish to suppress initial media, only providing media once the final, CLUE-controlled streams have been negotiated.

If the receiver of the initial offer is in the latter category and will be making the call CLUE-enabled with their SDP answer, they SHOULD reject all non-CLUE-controlled media lines that relate to the media they do not wish to receive in their SDP answer.

4.5.3. Processing the initial Offer/Answer negotiation

In the event that both offer and answer include a data channel "m" line with a mid value included in corresponding CLUE groups CLUE has been successfully negotiated and the call is now CLUE-enabled, otherwise the call is not CLUE enabled.

4.5.3.1. Successful CLUE negotiation

In the event of successful CLUE enablement of the call, devices MUST now begin negotiation of the CLUE channel, see [I-D.ietf-clue-datachannel] for negotiation details. If negotiation is successful, sending of CLUE protocol [I-D.presta-clue-protocol] messages can begin.

A CLUE-capable device MAY choose not to send media on the non-CLUE-controlled channels during the period in which control of the CLUE-controlled media lines is being negotiated. However, a CLUE-capable device MUST still be prepared to receive media on non-CLUE-controlled media lines that have been successfully negotiated as defined in [RFC3264].

If either side of the call wishes to add additional CLUE-controlled "m" lines to send or receive CLUE-controlled media they MAY now send

a SIP request with a new SDP offer. Note that if BUNDLE has been successfully negotiated and a Bundle Address Synchronization offer is required, the device to receive that offer SHOULD NOT generate a new SDP offer until it has received that BAS offer.

4.5.3.2. CLUE negotiation failure

In the event that the negotiation of CLUE fails and the call is not CLUE enabled in the initial offer/answer then CLUE is not in use in the call, and the CLUE-capable devices MUST either revert to non-CLUE behaviour or terminate the call.

4.5.4. Modifying the session

4.5.4.1. Adding and removing CLUE-controlled media

Subsequent offer/answer exchanges MAY add additional "m" lines for CLUE-controlled media; in most cases at least one additional exchange will be required before both sides have added all the encodings and ability to receive encodings that they desire. Devices MAY delay adding "a=recvonly" CLUE-controlled m-lines until after CLUE protocol negotiation completes - see Section 5 for recommendations.

Subsequent offer/answer exchanges MAY also deactive "m" lines for CLUE-controlled media.

Once CLUE media has been successfully negotiated devices SHOULD ensure that non-CLUE-controlled media is deactived in cases where it corresponds to CLUE-controlled media that has been successfully negotiated. Implementations can decide if they wish to disable non-CLUE controlled media once the call has been made CLUE enabled, or to wait until sending of the CLUE-controlled media has been successfully negotiated.

4.5.4.2. Enabling CLUE mid-call

A CLUE-capable device that receives an initial SDP offer from a non-CLUE device SHOULD include a new data channel "m" line and corresponding CLUE group in any subsequent offers it sends, to indicate that it is CLUE-capable.

If, in an ongoing non-CLUE call, one or both sides of the call add the CLUE data channel "m" line to their SDP and places the "mid" for that channel in corresponding CLUE groups then the call is now CLUE-enabled; negotiation of the data channel and subsequently the CLUE protocol begin.

4.5.4.3. Disabling CLUE mid-call

If, in an ongoing CLUE-enabled call, an SDP offer-answer negotiation completes in a fashion in which either the CLUE data channel was not successfully negotiated or one side did not include the data channel in a matching CLUE group then CLUE for this channel is disabled. In the event that this occurs, CLUE is no longer enabled and sending of all CLUE-controlled media associated with the corresponding CLUE group MUST stop.

Note that this is distinct to cases where the CLUE data channel fails or an error occurs on the CLUE protocol; see [I-D.presta-clue-protocol] for details of media and state preservation in this circumstance.

5. Interaction of CLUE protocol and SDP negotiations

Information about media streams in CLUE is split between two message types: SDP, which defines media addresses and limits, and the CLUE channel, which defines properties of capture devices available, scene information and additional constraints. As a result certain operations, such as advertising support for a new transmissible capture with associated stream, cannot be performed atomically, as they require changes to both SDP and CLUE messaging.

This section defines how the negotiation of the two protocols interact, provides some recommendations on dealing with intermediary stages in non-atomic operations, and mandates additional constraints on when CLUE-configured media can be sent.

5.1. Independence of SDP and CLUE negotiation

To avoid the need to implement interlocking state machines with the potential to reach invalid states if messages were to be lost, or be rewritten en-route by middle boxes, the state machines in SDP and CLUE operate independently. The state of the CLUE channel does not restrict when an implementation may send a new SDP offer or answer, and likewise the implementation’s ability to send a new CLUE ADVERTISEMENT or CONFIGURE message is not restricted by the results of or the state of the most recent SDP negotiation.

The primary implication of this is that a device may receive an SDP with a CLUE encoding it does not yet have capture information for, or receive a CLUE CONFIGURE message specifying a capture encoding for which the far end has not negotiated a media stream in SDP. An implementation that is

CLUE messages contain an <encID> which is used to identify a specific encoding or captureEncoding in SDP. The non-atomic nature of CLUE negotiation means that a sender may wish to send a new ADVERTISEMENT before the corresponding SDP message. As such the sender of the CLUE message MAY include an <encID> which does not currently match a CLUE-controlled "m" line label in SDP; A CLUE-capable implementation MUST not reject CLUE protocol messages that contain <encID> elements that do not match an id in SDP.

The current state of the CLUE participant or CLUE media provider/consumer state machines do not affect compliance with any of the normative language of [RFC3264]. That is, they MUST NOT delay an ongoing SDP exchange as part of a SIP server or client transaction; an implementation MUST NOT delay an SDP exchange while waiting for CLUE negotiation to complete or for a CONFIGURE message to arrive.

Similarly, a device in a CLUE-enabled call MUST NOT delay any mandatory state transitions in the CLUE participant or media provider/consumer state machines due to the presence or absence of an ongoing SDP exchange.

A device with the CLUE participant state machine in the ACTIVE state MAY choose not to move from CONF COMPLETED to PREPARING ADV (media provider state machine) or from READY TO CONF to TRYING (media consumer state machine) based on the SDP state. See [I-D.presta-clue-protocol] for CLUE state machine specifics. Similarly, a device MAY choose to delay initiating a new SDP exchange based on the state of their CLUE state machines.

5.2. Constraints on sending media

While SDP and CLUE message states do not impose constraints on each other, both impose constraints on the sending of media - CLUE-controlled media MUST NOT be sent unless it has been negotiated in both CLUE and SDP: an implementation MUST NOT send a specific CLUE capture encoding unless its most recent SDP exchange contains an active media channel for that encoding AND the far end has sent a CLUE CONFIGURE message specifying a valid capture for that encoding.

5.3. Recommendations for operating with non-atomic operations

CLUE-capable devices MUST be able to handle states in which CLUE messages make reference to EncodingIDs that do not match the most recently received SDP, irrespective of the order in which SDP and CLUE messages are received. While these mis-matches will usually be transitory a device MUST be able to cope with such mismatches remaining indefinitely. However, this document makes some recommendations on message ordering for these non-atomic transitions.

CLUE-capable devices SHOULD ensure that any inconsistencies between SDP and CLUE signalling are temporary by sending updated SDP or CLUE messages as soon as the relevant state machines and other constraints permit.

Generally, implementations that receive messages for which they have incomplete information SHOULD wait until they have the corresponding information they lack before sending messages to make changes related to that information. For instance, an implementation that receives a new SDP offer with three new "a=sendonly" CLUE "m" lines that has not received the corresponding CLUE ADVERTISEMENT providing the capture information for those streams SHOULD NOT include corresponding "a=recvonly" lines in its answer, but instead should make a new SDP offer when and if a new ADVERTISEMENT arrives with captures relevant to those encodings.

Because of the constraints of offer/answer and because new SDP negotiations are generally more 'costly' than sending a new CLUE message, implementations needing to make changes to both channels SHOULD prioritize sending the updated CLUE message over sending the new SDP message. The aim is for the recipient to receive the CLUE changes before the SDP changes, allowing the recipient to send their SDP answers without incomplete information, reducing the number of new SDP offers required.

6. Multiplexing of CLUE-controlled media using BUNDLE

6.1. Overview

A CLUE call may involve sending and/or receiving significant numbers of media streams. Conventionally, media streams are sent and received on unique ports. However, each seperate port used for this purpose may impose costs that a device wishes to avoid, such as the need to open that port on firewalls and NATs, the need to collect ICE candidates [RFC5245], etc.

The BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation] extension can be used to negotiate the multiplexing of multiple media lines onto a single 5-tuple for sending and receiving media, allowing devices in calls to another BUNDLE-supporting device to potentially avoid some of the above costs.

While CLUE-capable devices MAY support the BUNDLE extension for this purpose supporting the extension is not mandatory for a device to be CLUE-compliant.

6.2. Usage of BUNDLE with CLUE

This specification imposes no additional requirements or restrictions on the usage of BUNDLE when used with CLUE. There is no restriction on combining CLUE-controlled media lines and non-CLUE-controlled media lines in the same BUNDLE group or in multiple such groups. However, there are several steps an implementation may wish to ameliorate the cost and time requirements of extra SDP offer/answer exchanges between CLUE and BUNDLE.

6.2.1. Generating the Initial Offer

BUNDLE mandates that the initial SDP offer MUST use a unique address for each m-line with a non-zero port. Because CLUE implementations generarlly will not include CLUE-controlled media lines with the exception of the data channel CLUE devices that support large numbers of streams can avoid ever having to open large numbers of ports if they successfully negotiate BUNDLE.

6.2.2. Bundle Address Synchronization

When using BUNDLE the initial offerer may be mandated to send a Bundle Address Synchronisation offer. If the initial offerer also followed the recommendation of not including CLUE-controlled media lines in their offer, they MAY choose to include them in this subsequent offer. In this circumstance the BUNDLE specification recommends that the offerer does not "modify SDP parameters that could get the answerer to reject the BAS offer". Including new CLUE-controlled media lines using codecs and other attributes used in existing media lines should not increase the chance of the answerer rejecting the BAS offer; implementations should consider carefully before including new codecs or other new SDP attributes in these CLUE-controlled media lines.

6.2.3. Multiplexing of the data channel and RTP media

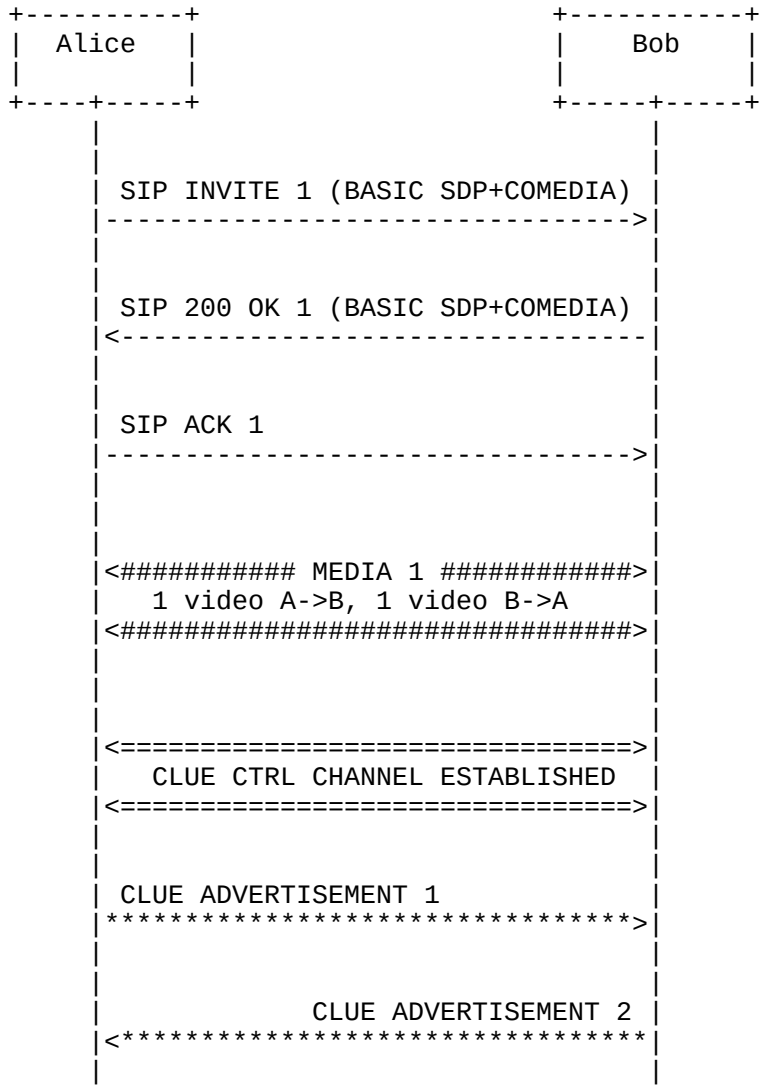
BUNDLE-supporting CLUE-capable devices MAY include the data channel in the same BUNDLE group as RTP media. In this case the device MUST be able to demultiplex the various transports - see section 7.2 of the BUNDLE draft [I-D.ietf-mmusic-sdp-bundle-negotiation]. If the BUNDLE group includes other protocols than the data channel transported via DTLS the device MUST also be able to differentiate the various protocols.

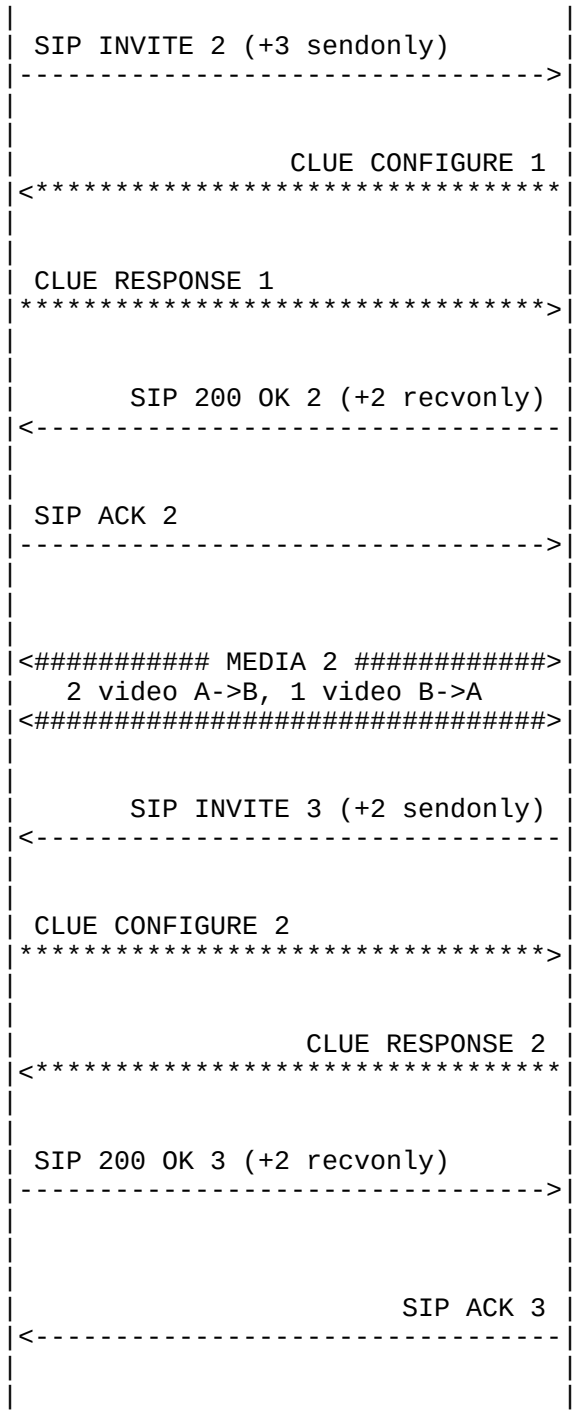
7. Example: A call between two CLUE-capable endpoints

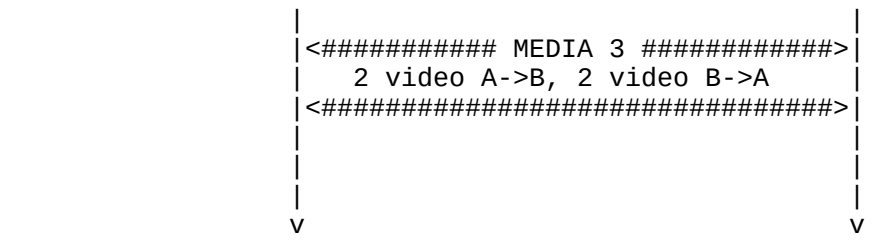
This example illustrates a call between two CLUE-capable endpoints.

Alice, initiating the call, is a system with three cameras and three screens. Bob, receiving the call, is a system with two cameras and two screens. A call-flow diagram is presented, followed by an summary of each message.

To manage the size of this section only video is considered, and SDP snippets only illustrate video 'm' lines. ACKs are not discussed. Note that BUNDLE is not in use.




SIP INVITE 2 (+3 sendonly)
CLUE CONFIGURE 1
CLUE RESPONSE 1
SIP 200 OK 2 (+2 recvonly)
SIP ACK 2
MEDIA 2
2 video A->B, 1 video B->A
SIP INVITE 3 (+2 sendonly)
CLUE CONFIGURE 2
CLUE RESPONSE 2
SIP 200 OK 3 (+2 recvonly)
SIP ACK 3



In INVITE 1, Alice sends Bob a SIP INVITE including in the SDP body the basilar audio and video capabilities ("BASIC SDP") and the information needed for opening a control channel to be used for CLUE protocol messages exchange, according to what is envisioned in the COMEDIA approach ("COMEDIA") for DTLS/SCTP channel [I-D.ietf-mmusic-sctp-sdp]. A snippet of the SDP showing the grouping attribute and the video m-line are shown below (mid 3 represents the CLUE channel):

```
...
a=group:CLUE 3
...
m=video 6002 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
a=sendrecv
a=mid:2
```

Bob responds with a similar SDP (200 OK 1); due to their similiarity no SDP snippet is shown here. Alice and Bob are each able to send a single audio and video stream (whether they choose to send this initial media before CLUE has been negotiated is implementation-dependent). This is illustrated as MEDIA 1.

With the successful initial O/A Alice and Bob are also free to negotiate the CLUE channel. Once this is successfully established CLUE negotiation can begin. This is illustrated as CLUE CHANNEL ESTABLISHED.

Alice now sends her CLUE Advertisement (ADVERTISEMENT 1). She advertises three static captures representing her three cameras. She also includes switched captures suitable for two- and one-screen systems. All of these captures are in a single capture scene, with suitable capture scene entries to tell Bob that he should either subscribe to the three static captures, the two switched capture view or the one switched capture view. Alice has no simultaneity

constraints, so includes all six captures in one simultaneous set. Finally, Alice includes an encoding group with three encoding IDs: "enc1", "enc2" and "enc3". These encoding ids aren't currently valid, but will match the next SDP offer she sends.

Bob received ADVERTISEMENT 1 but does not yet send a Configure message, because he has not yet received Alice's encoding information, so as yet he does not know if she will have sufficient resources to send him the two streams he ideally wants at a quality he is happy with.

Bob also sends his CLUE ADVERTISEMENT (ADVERTISEMENT 2). He advertises two static captures representing his cameras. He also includes a single composed capture for single-screen systems, in which he will composite the two camera views into a single video stream. All three captures are in a single capture scene, with suitable capture scene entries to tell Alice that she should either subscribe to the two static captures, or the single composed capture. Bob also has no simultaneity constraints, so includes all three captures in one simultaneous set. Bob also includes a single encoding group with two encoding IDs: "foo" and "bar".

Similarly, Alices receives ADVERTISEMENT 2 but does not yet send a CONFIGURE message, because she has not yet received Bob's encoding information.

Alice now sends INVITE 2. She maintains the sendrecv audio, video and CLUE m-lines, and she adds three new sendonly m-lines to represents the maximum three encodings she can send. Each of these m-lines has a label corresponding to one of the encoding ids from ADVERTISEMENT 1. Each also has its mid added to the grouping attribute to show they are controlled by the CLUE channel. A snippet of the SDP showing the grouping attribute and the video m-lines are shown below (mid 3 represents the CLUE channel):

```
...
a=group:CLUE 3 4 5 6
...
m=video 6002 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
a=sendrecv
a=mid:2
...
m=video 6004 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016
a=sendonly
a=mid:4
a=label:enc1
m=video 6006 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016
a=sendonly
a=mid:5
a=label:enc2
m=video 6008 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016
a=sendonly
a=mid:6
a=label:enc3
```

Bob now has all the information he needs to decide which streams to configure. As such he now sends CONFIGURE 1. This requests the pair of switched captures that represent Alice's scene, and he configures them with encoder ids "enc1" and "enc2". This also serves as an ack for Alice's ADVERTISMENT 1.

Alice receives Bob's message CONFIGURE 1 and sends RESPONSE 1 to ack its receptions. She does not yet send the capture encodings specified, because at this stage Bob hasn't negotiated the ability to receive these streams in SDP.

Bob now sends his SDP answer as part of 200 OK 2. Alongside his original audio, video and CLUE m-lines he includes two active recvonly m-lines and a zeroed m-line for the third. He adds their mid values to the grouping attribute to show they are controlled by the CLUE channel. A snippet of the SDP showing the grouping attribute and the video m-lines are shown below (mid 100 represents the CLUE channel):

```
...
a=group:CLUE 11 12 100
...
m=video 58722 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
a=sendrecv
a=mid:10
...
m=video 58724 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
a=recvonly
a=mid:11
m=video 58726 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
a=recvonly
a=mid:12
m=video 0 RTP/AVP 96
```

On receiving 200 OK 2 from Bob Alice is now able to send the two streams of video Bob requested - this is illustrated as MEDIA 2.

The constraints of offer/answer meant that Bob could not include his encoder information as new m-lines in 200 OK 2. As such Bob now sends INVITE 3 to generate a new offer. Along with all the streams from 200 OK 2 Bob also includes two new sendonly streams. Each stream has a label corresponding to the encoding ids in his ADVERTISEMENT 2 message. He also adds their mid values to the grouping attribute to show they are controlled by the CLUE channel. A snippet of the SDP showing the grouping attribute and the video m-lines are shown below (mid 100 represents the CLUE channel):

```
...
a=group:CLUE 11 12 13 14 100
...
m=video 58722 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
a=sendrecv
a=mid:10
...
m=video 58724 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
a=recvonly
a=mid:11
m=video 58726 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
a=recvonly
a=mid:12
m=video 0 RTP/AVP 96
m=video 58728 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016
a=sendonly
a=label:foo
a=mid:13
m=video 58730 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016
a=sendonly
a=label:bar
a=mid:14
```

Having received this Alice now has all the information she needs to send CONFIGURE 2. She requests the two static captures from Bob, to be sent on encodings "foo" and "bar".

Bob receives Alice's message CONFIGURE 2 and sends RESPONSE 2 to ack its receptions. Bob does not yet send the capture encodings specified, because Alice hasn't yet negotiated the ability to receive these streams in SDP.

Alice now sends 200 OK 3, matching two recvonly m-lines to Bob's new sendonly lines. She includes their mid values in the grouping attribute to show they are controlled by the CLUE channel. A snippet of the SDP showing the grouping attribute and the video m-lines are shown below (mid 3 represents the CLUE channel):

```
...
a=group:CLUE 3 4 5 7 8
...
m=video 6002 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
a=sendrecv
a=mid:2
...
m=video 6004 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016
a=sendonly
a=mid:4
a=label:enc1
m=video 6006 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016
a=sendonly
a=mid:5
a=label:enc2
m=video 0 RTP/AVP 96
m=video 6010 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
a=recvonly
a=mid:7
m=video 6012 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
a=recvonly
a=mid:8
```

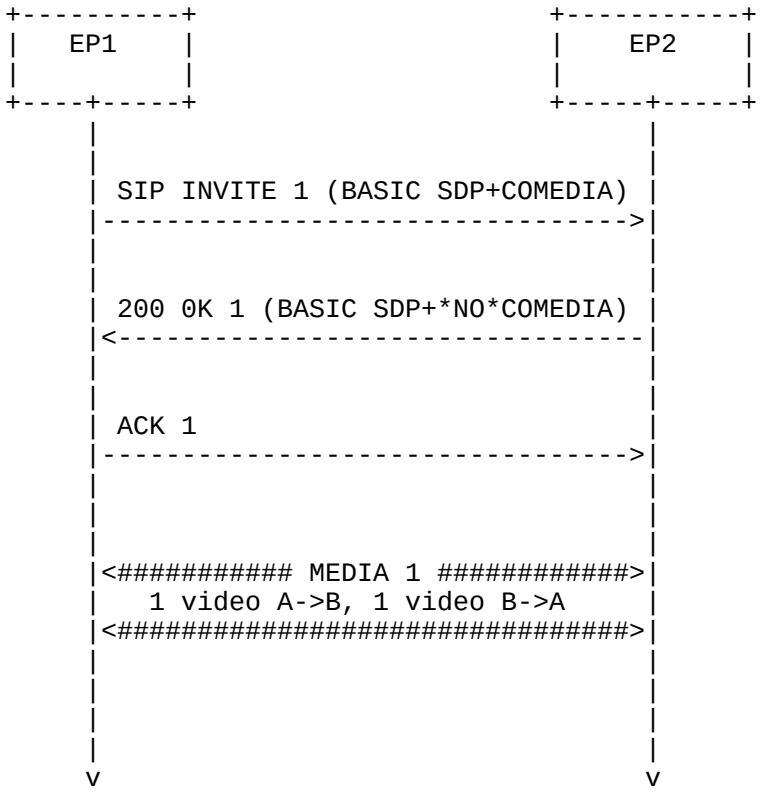
Finally, on receiving 200 OK 3 Bob is now able to send the two streams of video Alice requested - this is illustrated as MEDIA 3.

Both sides of the call are now sending multiple video streams with their sources defined via CLUE negotiation. As the call progresses either side can send new ADVERTISEMENT or CONFIGURE or new SDP negotiation to add, remove or change what they have available or want to receive.

8. Example: A call between a CLUE-capable and non-CLUE endpoint

In this brief example Alice is a CLUE-capable endpoint making a call to Bob, who is not CLUE-capable, i.e., it is not able to use the CLUE

protocol.



In INVITE 1, Alice sends Bob a SIP INVITE including in the SDP body the basilar audio and video capabilities ("BASIC SDP") and the information needed for opening a control channel to be used for CLUE protocol messages exchange, according to what is envisioned in the COMEDIA approach ("COMEDIA") for DTLS/SCTP channel [I-D.ietf-mmusic-sctp-sdp]. A snippet of the SDP showing the grouping attribute and the video m-line are shown below (mid 3 represents the CLUE channel):

```
...
a=group:CLUE 3
...
m=video 6002 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600
a=sendrecv
a=mid:2
```

Bob is not CLUE capable, and hence does not recognize the "CLUE" semantic for the grouping attribute, not does he support the CLUE channel. He responds with an answer with audio and video, but with the CLUE channel zeroed.

From the lack of the CLUE channel Alice understands that Bob does not support CLUE, or does not wish to use it. Both sides are now able to send a single audio and video stream to each other. Alice at this point begins to send her fallback video: in this case likely a switched view from whichever camera shows the current loudest participant on her side.

9. CLUE requirements on SDP O/A

The current proposal calls for a new "CLUE" semantic for the SDP Grouping Framework [RFC5888].

10. SIP Signaling

The current proposal calls for a "sip.clue" media feature tag.

11. CLUE over RTCWEB

We may want to rule this out of scope for now. But we should be thinking about this.

12. Open Issues

Here are issues pertinent to signaling that need resolution. Resolution will probably result in changes somewhere in this document, but may also impact other documents.

- The current method for expressing encodings in SDP limits the parameters available when describing H264 encoder capabilities to those defined in Table 6 in [RFC6184]

13. What else?

14. Acknowledgements

The team focusing on this draft consists of: Roni Even, Rob Hansen, Christer Holmberg, Paul Kyzivat, Simon Pietro-Romano, Roberta Presta.

Christian Groves and Jonathon Lennox have contributed detailed comments and suggestions.

The author list should be updated as people contribute substantial text to this document.

15. IANA Considerations

TBD

16. Security Considerations

TBD

17. Change History

-02: Revision by Rob Hansen

* Added section on not accepting non-CLUE-controlled "m" lines in the initial answer when CLUE is to be negotiated.
* Removed previous language attempting to describe media restrictions for CLUE-controlled "m" lines that had not been configured, and replaced it with much more accurate ’treat as "a=inactive" was set’.
* Made label element mandatory for CLUE-controlled media (was previously "SHOULD include", but there didn’t seem a good reason for this - anyone wishing to include the "m" line but not immediately use it in CLUE can simply leave it out of the <encodingIDList>.)
* Added a section on the specifics of relating encodings in SDP to <encID> elements in the CLUE protocol, including the fact that both ADVERTISMENTS and CONFIGURE messages reference the *encoding* (eg, in the CONFIGURE case the sender of the CONFIGURE message includes the labels of the recipient’s "m" lines as their <encID> contents).
* Minor revisions to the section on complying with normative SDP/CLUEstate machine language to clarify that these were not new normative language, merely that existing normative language

still applies.
* Removed appendices which previously contained information to be transferred to the protocol and data channel drafts. Removed other text that discussed alternatives to the current approach.
* Cleaned up some 'todo' text.

-01: Revision by Rob Hansen
* Revised terminology - removed the term 'CLUE-enabled' device as insufficiently distinct from 'CLUE-capable' and instead added a term for 'CLUE-enabled' calls.
* Removed text forbidding RTCP and instead added text that ICE/DTLS negotiation for CLUE controlled media must be done as normal irrespective of CLUE negotiation.
* Changed 'sip.telepresence' to 'sip.clue' and 'TELEPRESENCE' grouping semantic back to CLUE.
* Made it mandatory to have exactly one mid corresponding to a data channel in a CLUE group
* Forbade having multiple CLUE groups unless a specification for doing so is published
* Refactored SDP-related text; previously the encoding information had been in the "initial offer" section despite the fact that we recommend that the initial offer doesn't actually include any encodings. I moved the specifications of encodings and how they're received to an earlier, seperate section.
* Added text on how the state machines in CLUE and SDP are allowed to affect one another, and further recommendations on how a device should handle the sending of CLUE and SDP changes.

-00: Revision by Rob Hansen
* Submitted as -00 working group document

draft-kyzivat-08: Revisions by Rob Hansen
* Added media feature tag for CLUE support ('sip.telepresence')
* Changed grouping semantic from 'CLUE' to 'TELEPRESENCE'
* Restructured document to be more centred on the grouping semantic and its use with O/A
* Lots of additional text on usage of the grouping semantic
* Stricter definition of CLUE-controlled m lines and how they work
* Some additional text on defining what happens when CLUE supports is added or removed
* Added details on when to not send RTCP for CLUE-controlled "m" lines.
* Added a section on using BUNDLE with CLUE
* Updated data channel references to point at new WG document rather than indivual draft

draft-kyzivat-07: Revisions by Rob Hansen
* Removed the text providing arguments for encoding limits being in SDP and encoding groups in the CLUE protocol in favor of the specifics of how to negotiate encodings in SDP

* Added normative language on the setting up of a CLUE call, and added sections on mid-call changes to the CLUE status.
* Added references to [I-D.ietf-clue-datachannel] where appropriate.
* Added some terminology for various types of CLUE and non-CLUE states of operation.
* Moved language related to topics that should be in [I-D.ietf-clue-datachannel] and [I-D.presta-clue-protocol], but that has not yet been resolved in those documents, into an appendix.

draft-kyzivat-06: Revisions by Rob Hansen

* Removed CLUE message XML schema and details that are now in draft-presta-clue-protocol
* Encoding limits in SDP section updated to note that this has been investigated and discussed and is the current working assumption of the WG, though consensus has not been fully achieved.
* A section has also been added on the current mandation of unidirectional "m"-lines.
* Updated CLUE messaging in example call flow to match draft-presta-clue-protocol-03

draft-kyzivat-05: Revisions by pkyzivat:

* Specified versioning model and mechanism.
* Added explicit response to all messages.
* Rearranged text to work with the above changes. (Which rendered diff almost useless.)

draft-kyzivat-04: Revisions by Rob Hansen: ???

draft-kyzivat-03: Revisions by pkyzivat:

* Added a syntax section with an XML schema for CLUE messages. This is a strawhorse, and is very incomplete, but it establishes a template for doing this based on elements defined in the data model. (Thanks to Roberta for help with this!)
* Did some rewording to fit the syntax section in and reference it.
* Did some relatively minor restructuring of the document to make it flow better in a logical way.

draft-kyzivat-02: A bunch of revisions by pkyzivat:

* Moved roberta’s call flows to a more appropriate place in the document.
* New section on versioning.
* New section on NAK.
* A couple of possible alternatives for message acknowledgment.
* Some discussion of when/how to signal changes in provider state.
* Some discussion about the handling of transport errors.
* Added a change history section.

These were developed by Lennard Xiao, Christian Groves and Paul, so added Lennard and Christian as authors.

draft-kyzivat-01: Updated by roberta to include some sample call flows.
draft-kyzivat-00: Initial version by pkyzivat. Established general outline for the document, and specified a few things thought to represent wg consensus.

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.

[I-D.ietf-clue-framework] Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-16 (work in progress), June 2014.

[I-D.presta-clue-data-model-schema] Presta, R. and S. Romano, "An XML Schema for the CLUE data model", draft-presta-clue-data-model-schema-03 (work in progress), March 2013.

[I-D.presta-clue-protocol] Presta, R. and S. Romano, "CLUE protocol", draft-presta-clue-protocol-04 (work in progress), May 2014.

[I-D.ietf-clue-datachannel] Holmberg, C., "CLUE Protocol Data Channel", draft-ietf-clue-datachannel-00 (work in progress), March 2014.

[I-D.groves-clue-latent-config] Groves, C., Yang, W., and R. Even, "CLUE and latent configurations", draft-groves-clue-latent-config-00 (work in progress), January 2014.

[I-D.ietf-mmusic-sctp-sdp] Loreto, S. and G. Camarillo, "Stream Control Transmission Protocol (SCTP)-Based Media Transport in the Session Description Protocol (SDP)", draft-ietf-mmusic-sctp-sdp-06 (work in progress), February 2014.

[I-D.tuexen-tsvwg-sctp-dtls-encaps] Jesup, R., Loreto, S., Stewart, R., and M. Tuexen, "DTLS

Encapsulation of SCTP Packets for RTCWEB", draft-tuexen-tsvwg-sctp-dtls-encaps-01 (work in progress), July 2012.

[RFC4574] Levin, O. and G. Camarillo, "The Session Description Protocol (SDP) Label Attribute", RFC 4574, August 2006.

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

18.2. Informative References

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

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

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC4353] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353, February 2006.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.

[RFC6184] Wang, Y., Even, R., Kristensen, T., and R. Jesup, "RTP Payload Format for H.264 Video", RFC 6184, May 2011.

[I-D.even-clue-sdp-clue-relation] Even, R., "Signalling of CLUE and SDP offer/answer", draft-even-clue-sdp-clue-relation-01 (work in progress), October 2012.

[I-D.even-clue-rtp-mapping] Even, R. and J. Lennox, "Mapping RTP streams to CLUE media captures", draft-even-clue-rtp-mapping-05 (work in progress), February 2013.

[I-D.hansen-clue-sdp-interaction] Hansen, R., "SDP and CLUE message interactions", draft-hansen-clue-sdp-interaction-01 (work in progress), February 2013.

[I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-07 (work in progress), April 2014.

Authors' Addresses

Paul Kyzivat
Huawei

Email: pkyzivat@alum.mit.edu

Lennard Xiao
Huawei

Email: lennard.xiao@huawei.com

Christian Groves
Huawei

Email: Christian.Groves@nteczone.com

Robert Hansen
Cisco Systems

Email: rohanse2@cisco.com