

PERC
Internet-Draft
Intended status: Informational
Expires: April 18, 2016

C. Groves, Ed.
W. Yang
R. Even
Huawei
October 16, 2015

Usage of CLUE with PERC
draft-groves-perc-clue-00

Abstract

This document provides an initial discussion of the relationship between PERC and CLUE. It seeks to identify any potential impacts or/and enhancement to the way that CLUE is used in the PERC architecture.

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 April 18, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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	2
2. Requirements Language	3
3. CLUE Background	3
4. CLUE Relation to PERC	6
4.1. Topology	6
4.2. Media manipulation	7
4.3. Privacy	7
4.4. Encodings	8
4.5. Mapping RTP streams to CLUE media captures	8
4.6. Others?	9
5. Potential CLUE enhancements	9
5.1. Encrypted CLUE information	9
5.2. Others?	11
6. Summary	11
7. Acknowledgements	11
8. IANA Considerations	11
9. Security Considerations	11
10. References	11
10.1. Normative References	12
10.2. Informative References	12
Authors' Addresses	13

1. Introduction

OThe PERC working charter specifically mentions that the solution for PERC should:

"be implementable by both SIP (RFC3261) and WebRTC endpoints [I-D.ietf-rtcweb-overview]. How telepresence endpoints using the protocols defined in the CLUE working group could utilize the defined security solution needs to be considered. However, it is acknowledged that limitations may exist, resulting in restricted functionality or need for additional adaptations of the CLUE protocols."

It also indicates that work for documenting the model for integrating PERC with based with the establishment of CLUE conferences needs to be performed.

This draft provides some initial information to address both these areas.

2. Requirements Language

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] 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. CLUE Background

The CLUE protocol framework [I-D.ietf-clue-framework] effectively is a means of sending metadata about media captures and encodings between a Providing Endpoint and a Consuming Endpoint. The CLUE protocol is transmitted using a WebRTC Datachannel [I-D.ietf-clue-datachannel] meaning that any SRTP based mechanisms for encrypting this metadata cannot be used.

The information that can be sent regarding media captures is summarized below:

- Spatial information, including point of capture, point on line of capture, area of capture, mobility of capture and audio capture sensitivity pattern;
- Descriptive information, including a human readable description, indication of presentation, field of view type and language;
- Person information, including the role of the person and xCard description;
- Miscellaneous information, including whether text is embedded or a relation to other captures.

It is possible for providers through the Multi-Content Capture (MCC) mechanism to provide the information about the individual contributing sources. It can provide the switching policy as well as synchronization information.

Information about the overall "Scene" may also be provided including views, human readable descriptions, xCard and scale information.

Using the CLUE protocol information the Consuming endpoint can then choose what media captures and encodings that it would like to receive through the use of CLUE and SIP/SDP signalling. Media is typically provided through SRTP. Figure 2 / [I-D.ietf-clue-framework] highlights the basic call flow. Figure 1 below provides a copy of this flow.

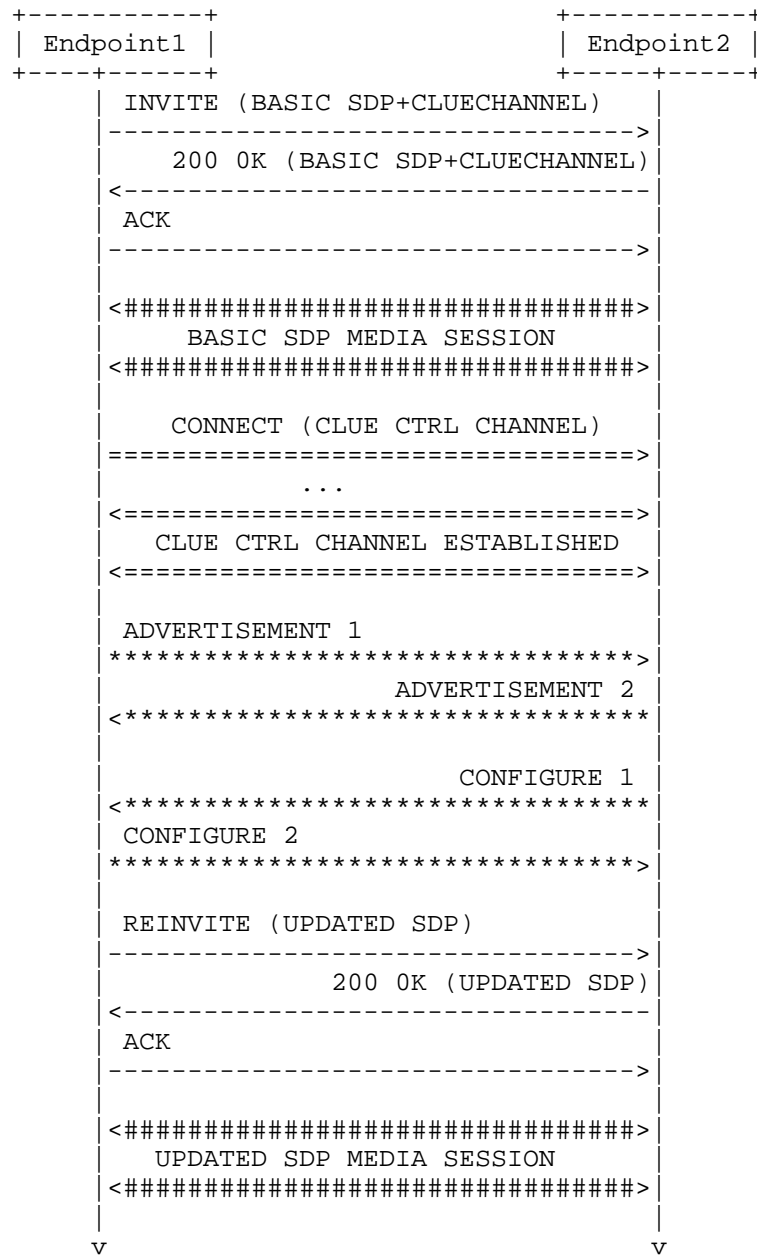


Figure 1: CLUE Basic Information Flow

Whilst CLUE is a point to point protocol it may be used in conferences containing multipoint control units (MCU)s. In this

scenario the MCU acts as an aggregation point for CLUE information. That is the MCU receives ADVERTISEMENT messages received from multiple endpoints before deciding on the contents of the ADVERTISEMENT that it wishes to send. Likewise the MCU will use received CONFIGURE messages to decide what the contents of its CONFIGURE messages will be. In doing so the MCU may apply any local policy / provisioning information to its decisions. Figure 2 illustrates this CLUE signalling. The SIP/SDP signalling is omitted for brevity.

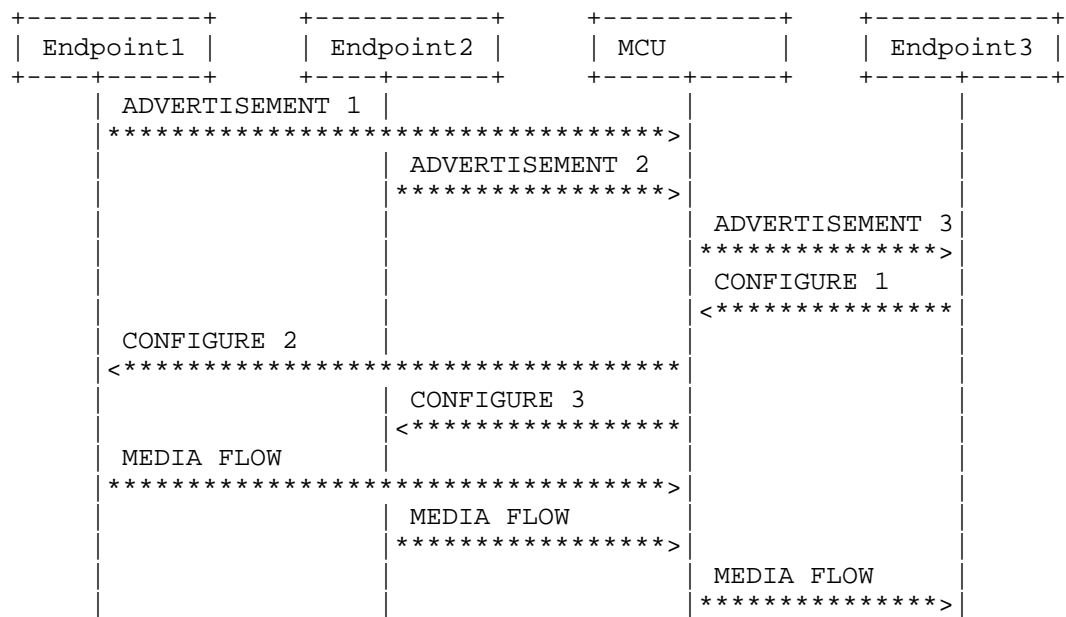


Figure 2: CLUE MCU Flow

Figure 2 shows a unidirectional media flow to Endpoint3. A bi-directional media flow would be enabled by Endpoint3 providing an ADVERTISEMENT to the MCU and the MCU providing ADVERTISEMENTS to Endpoint1 and Endpoint2. Endpoint1 and Endpoint2 would then also send CONFIGURE messages to the MCU and the MCU would send a CONFIGURE message to Endpoint3.

Thus the selection of a particular Media Capture and Encoding (Capture Encoding) by the endpoints drives what topology occurs at the MCU. However there is a caveat. The MCU could apply filtering of the CLUE metadata to provide a sub-set of the data or append its own data. For example it may decide that rather than offer the

source audio from Endpoint1 and Endpoint2 as individual streams, it will offer a mix of these two sources, as individual streams.

4. CLUE Relation to PERC

As detailed in the charter, the goal of the PERC WG is to:

"...work on a solution that enables centralized SRTP-based conferencing, where the central device distributing the media is not required to be trusted with the keys to decrypt the participants' media. The media must be kept confidential and authenticated between an originating endpoint and the explicitly allowed receiving endpoints or other devices. The meta information provided to the central device is to be limited to the minimal required for it to perform its function to preserve the conference participant's privacy."

As described above CLUE largely provides metadata (or meta information) so the task is to identify the minimal set of CLUE data required for CLUE to still work. It also needs to be considered the limited functionality of a media distribution device (MDD) as compared to an MCU.

In broad terms the initial PERC drafts propose a solution where there are two sets of encryption keys, one for the end-to-end (e2e) session and another for the transport connection (i.e. between the endpoint and MDD). SRTP extensions are required to carry the e2e encrypted data. The concept of a key management function (KMF) is also introduced which receives information about the call and the endpoints (as per 8.1/[I-D.jones-perc-private-media-framework]). The KMF is the element that the endpoints trust it provides cryptographic keys and authenticates media content. See 6.1 / [I-D.jones-perc-private-media-reqts].

So far only SRTP media has been considered by PERC. As the MDD does not have access to the un-encrypted media stream it can only provide switching topologies (e.g. Media Switch, Selective Forwarding Unit, Transport Translator/ Transport Relay?, [I-D.ietf-avtcore-rtp-topologies-update]).

These aspects have some implications on the use of CLUE.

4.1. Topology

As noted in the background section the selection of particular captures and encodings by endpoints effectively dictates what topology occurs at the MCU/MDD. Therefore where a CLUE enabled MDD receives an indication that an encoding requires the use of PERC, the

MDD must ensure that in any subsequent ADVERTISEMENTS and SIP/SDP offers it sends, that the capture encoding is an unmixed local source (i.e. doesn't use a MCC indicating a MDD local composition of remote sources). This would be a mismatch in capabilities as the MDD is unable to mix SRTP streams.

Whilst the MDD may utilise the MCC mechanism to indicate that a particular capture encoding may represent multiple sources, the MaxCaptures attribute (section 7.2.1.1/[I-D.ietf-clue-framework]) should be set to ≤ 1 or 1 to indicate that only switching is used. Other MaxCapture values indicate the potential use of composed (thus mixed) capture encodings.

A MCC Policy attribute (section 7.2.1.2/[I-D.ietf-clue-framework]) may also be included. It allows the indication of a "SoundLevel" policy that indicates that the content of the capture encoding is determined by a sound level detection algorithm. As a PERC enabled MDD cannot access the SRTP media the "SoundLevel" policy should not be used unless the endpoint indicates the use of an unencrypted or hop by hop mechanism (e.g. utilising [RFC6464]) for sound level detection.

4.2. Media manipulation

Given that the PERC enabled MDD cannot access the encrypted media, it cannot filter possible media content. For example an endpoint may indicate that the media capture contains embedded text (clause 7.1.1.13/[I-D.ietf-clue-framework]) information. It has no mechanism to filter out (e.g. by removing part of the image, or text signaled associated with audio) or to confirm text is being sent. Therefore the MDD can either only remove the capture from being ADVERTISED or pass the embedded text attribute without modification.

A CLUE enabled MDD has the ability of adding its own captures and encodings to ADVERTISEMENTS. PERC enabled consumers should determine if the encoding associated with the advertised captures contains the correct key/fingerprint information as distributed by the KMF before requesting the capture encoding via a CONFIGURE. This is a similar consideration as for non-CLUE endpoint responding with an SDP Answer.

4.3. Privacy

The CLUE framework allows the sending of potentially private information to the MCU. Participant and endpoint information via the xCard [RFC6351] format may be provided. xCard can contain address, contact, company, images and audio information. Whilst this information does not compromise the encrypted media it does provide information about the persons generating it.

CLUE also allows the definition of extensions so there may be proprietary extensions that may also contain potentially sensitive information.

As indicated in the PERC WG charter meta information provided to the central device is to be limited to the minimal required for the MDD to perform its function. This may potentially result in an CLUE endpoint significantly reducing the amount of metadata it sends in ADVERTISEMENTS. This would result in decreased information for CONSUMERS to decide which captures to consumer. This may lead to a decreased telepresence user experience.

4.4. Encodings

CLUE itself carries little encoding information other than encoding groups with indicate which encodings are linked (and the maximum bit rate) and encodingIDs of the individual encodings. The encodingIDs provide a link to the actual encoding information provided through SIP/SDP. The SDP utilizes the "a=group" and "a=mid" mechanism to reference the CLUE encodingIDs thus providing a linkage between CLUE and SDP.

It is expected that any indication of the use of PERC for SRTP streams will be signaled through SDP. Therefore a CLUE enabled endpoint is not required to change any CLUE based encoding information to use PERC.

4.5. Mapping RTP streams to CLUE media captures

In order to associate RTP media with a particular CLUE capture encoding [I-D.ietf-clue-rtp-mapping] defines a RTP header extension and a RTCP SDES item both containing a CaptureID. The draft indicates for mapping an RTP stream to a specific MC in the MCC the CLUE the media sender MUST send for MCC the captureID in the RTP header and as a RTCP SDES message.

If an MDD produces or modifies MCCs (in particular the individual source CaptureIDs) as per section 4.1 above, then it may need to potentially modify the received source RTP/RTCP captureIDs to match the CLUE MCC before sending RTP/RTCP. In the case of voice activated switching, the MDD should also send the relevant RTP/RTCP captureID.

Therefore any PERC solution should ensure that the MDD may have access to and the ability to send RTP/RTCP captureID.

4.6. Others?

TBD

5. Potential CLUE enhancements

CLUE has a defined extension mechanism (see section 8/[ID.ietf-clue-protocol]). The use of any enhancements related to PERC could be negotiated through this mechanism.

5.1. Encrypted CLUE information

In order to limit the amount of metadata available to the MDD but still allowing the full use of CLUE, CLUE could be enhanced to carry encrypted data that is associated with a capture/scene but is not available to an MDD. This would be similar to the proposed solution for SRTP. The KMF could be enhanced to provide keys to the endpoints to access this CLUE encrypted data to make decisions on which capture encodings to CONFIGURE.

In a PERC environment the endpoints are responsible for stream selection and any composition and thus they should have access to the full capture and scene metadata provided by the other endpoints in a conference. A MDD that switches streams doesn't need access to this metadata as it should not make decisions regarding the forwarding of streams based on the content/characteristics of the stream. Accordingly the MDD only strictly needs a CaptureID and the encoding information in order to switch streams. CLUE capture attributes, capture scene, simultaneous set and people information may be encrypted and passed through the MDD. This is due to the fact that a CONFIGURE only contains a CaptureID and an associated EncodingID. It's the CONFIGURE message that determine which capture encoding an endpoint sends.

The syntax below in figure 3 provides a conceptual illustration of the clear and encrypted parts of a CLUE ADVERTISEMENT utilising the example from section 10.1 / [ID.ietf-clue-protocol]:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<advertisement xmlns="urn:ietf:params:xml:ns:clue-protocol"
  xmlns:ns2="urn:ietf:params:xml:ns:clue-info"
  xmlns:ns3="urn:ietf:params:xml:ns:vcard-4.0"
  protocol="CLUE" v="0.4">
  <clueId>Napoli</clueId>
  <sequenceNr>45</sequenceNr>
  <mediaCaptures>
    <ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="ns2:videoCaptureType" captureID="AC0" mediaType="video">
```

```

        <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
        <ns2:encGroupIDREF>EG1</ns2:encGroupIDREF>
        /***** Encrypted contents *****/
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" mediaType="video" captureID="VC0">
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
  /***** Encrypted contents *****/
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" mediaType="video" captureID="VC1">
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
  /***** Encrypted contents *****/
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" mediaType="video" captureID="VC3">
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
  /***** Encrypted contents *****/
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" mediaType="video" captureID="VC4">
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
  /***** Encrypted contents *****/
</mediaCaptures>
<encodingGroups>
  <ns2:encodingGroup encodingGroupID="EG0">
    <ns2:maxGroupBandwidth>600000</ns2:maxGroupBandwidth>
    <ns2:encodingIDList>
      <ns2:encID>ENC1</ns2:encID>
      <ns2:encID>ENC2</ns2:encID>
      <ns2:encID>ENC3</ns2:encID>
    </ns2:encodingIDList>
  </ns2:encodingGroup>
  <ns2:encodingGroup encodingGroupID="EG1">
    <ns2:maxGroupBandwidth>300000</ns2:maxGroupBandwidth>
    <ns2:encodingIDList>
      <ns2:encID>ENC4</ns2:encID>
      <ns2:encID>ENC5</ns2:encID>
    </ns2:encodingIDList>
  </ns2:encodingGroup>
</encodingGroups>
<captureScenes>
  /***** Encrypted contents *****/
</captureScenes>
<simultaneousSets>

```

```
      /***** Encrypted contents *****/
    </simultaneousSets>
    <people>
      /***** Encrypted contents *****/
    </people>
  </advertisement>
```

Figure 3: Encrypted CLUE Advertisement

The downside of this approach is that the MDD effectively becomes unable to offer its own switched streams as multiple content captures. Whilst in theory it could offer its own MCCs utilising the unencrypted CaptureIDs, it has little metadata to decide which streams are related in order to provide synchronised switching. Therefore it could be recommended that information such as the capture area (which is unlikely to be sensitive) should be passed in the clear (unencrypted) to allow the MDD to distinguish that the captures cover different parts of the same scene. In this case the MDD could provide a MCC.

5.2. Others?

TBD

6. Summary

This draft provides a discussion of the relationship between CLUE and PERC and the potential impacts to CLUE when used with PERC streams.

7. Acknowledgements

This template was derived from an initial version written by Pekka Savola and contributed by him to the xml2rfc project.

8. IANA Considerations

None.

9. Security Considerations

This draft is about the privacy and security implications of using CLUE in a PERC environment.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

10.2. Informative References

- [I-D.ietf-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update-10 (work in progress), July 2015.
- [I-D.ietf-clue-datachannel]
Holmberg, C., "CLUE Protocol data channel", draft-ietf-clue-datachannel-10 (work in progress), September 2015.
- [I-D.ietf-clue-framework]
Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-23 (work in progress), September 2015.
- [I-D.ietf-clue-rtp-mapping]
Even, R. and J. Lennox, "Mapping RTP streams to CLUE media captures", draft-ietf-clue-rtp-mapping-04 (work in progress), March 2015.
- [I-D.jones-perc-private-media-framework]
Jones, P., Ismail, N., and D. Benham, "A Solution Framework for Private Media in Privacy Enhanced RTP Conferencing", draft-jones-perc-private-media-framework-01 (work in progress), October 2015.
- [I-D.jones-perc-private-media-reqts]
Jones, P., Ismail, N., Benham, D., Buckles, N., Mattsson, J., and R. Barnes, "Private Media Requirements in Privacy Enhanced RTP Conferencing", draft-jones-perc-private-media-reqts-00 (work in progress), July 2015.
- [RFC6351] Perreault, S., "xCard: vCard XML Representation", RFC 6351, DOI 10.17487/RFC6351, August 2011, <<http://www.rfc-editor.org/info/rfc6351>>.

[RFC6464] Lennox, J., Ed., Iovov, E., and E. Marocco, "A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", RFC 6464, DOI 10.17487/RFC6464, December 2011, <<http://www.rfc-editor.org/info/rfc6464>>.

Authors' Addresses

Christian Groves (editor)
Huawei
Melbourne
Australia

Email: Christian.Groves@nteczone.com

Weiwei Yang
Huawei
P.R.China

Email: tommy@huawei.com

Roni Even
Huawei
Tel Aviv
Isreal

Email: roni.even@mail01.huawei.com

CLUE Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 24, 2015

R. Presta
S. Romano
University of Napoli
April 22, 2015

CLUE protocol
draft-ietf-clue-protocol-04

Abstract

The CLUE protocol is an application protocol conceived for the description and negotiation of a CLUE telepresence session. The design of the CLUE protocol takes into account the requirements and the framework defined, respectively, in [I-D.ietf-clue-framework] and [RFC7262]. The companion document [I-D.ietf-clue-signaling] delves into CLUE signaling details, as well as on the SIP/SDP session establishment phase. CLUE messages flow upon the CLUE data channel, based on reliable and ordered SCTP over DTLS transport, as described in [I-D.ietf-clue-datachannel]. Message details, together with the behavior of CLUE Participants acting as Media Providers and/or Media Consumers, are herein discussed.

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 October 24, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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	4
2. Terminology	4
3. Overview of the CLUE protocol	5
4. Protocol messages	7
4.1. OPTIONS	9
4.2. OPTIONS RESPONSE	11
4.3. ADVERTISEMENT	12
4.4. ADVERTISEMENT ACKNOWLEDGEMENT	13
4.5. CONFIGURE	14
4.6. CONFIGURE RESPONSE	14
4.7. Response codes and reason strings	15
5. Protocol state machines	17
6. CLUE Participant's state machine	17
6.1. Media Provider's state machine	19
6.2. Media Consumer's state machine	22
7. Versioning	25
8. Extensions and options	26
9. XML Schema	27
10. Examples	31
10.1. Simple ADV	31
10.2. ADV with MCCs	37
11. IANA Considerations	44
11.1. URN Sub-Namespace Registration	44
11.2. XML Schema registration	45
11.3. MIME Media Type Registration for 'application/clue+xml'	45
11.4. DNS Registrations	46
11.4.1. Application Service tag	46
11.4.2. Application Protocol tag	46
11.5. CLUE Protocol Registry	47
11.5.1. CLUE Message Types	47
11.5.2. CLUE Response Codes	48
12. Diff with draft-ietf-clue-protocol-03	49
13. Diff with draft-ietf-clue-protocol-02	49
14. Diff with draft-ietf-clue-protocol-01	50
15. Diff with draft-ietf-clue-protocol-00	50
16. Diff with draft-presta-clue-protocol-04	51
17. Diff with draft-presta-clue-protocol-03	51
18. Diff with draft-presta-clue-protocol-02	51
19. Acknowledgments	51
20. References	51
20.1. Normative References	51
20.2. Informative References	52

1. Introduction

The CLUE protocol is an application protocol used by two CLUE Participants to enhance the experience of a multimedia telepresence session. The main goals of the CLUE protocol are:

1. enabling a MP to properly announce its current telepresence capabilities to a MC in terms of available media captures, groups of encodings, simultaneity constraints and other information envisioned in [I-D.ietf-clue-framework];
2. enabling a MC to request the desired multimedia streams from the offering MP.

CLUE-capable endpoints are connected by means of the CLUE data channel, an SCTP over DTLS channel which is opened and established as described in [I-D.ietf-clue-signaling] and [I-D.ietf-clue-datachannel]. CLUE protocol messages flowing upon such a channel are detailed in this document, both syntactically and semantically.

In Section 3 we provide a general overview of the CLUE protocol. CLUE protocol messages are detailed in Section 4. The CLUE Participant state machine is introduced in Section 5. Versioning and extensions are discussed in Section 7 and Section 8, respectively. The XML schema defining the CLUE messages is reported in Section 9.

2. Terminology

This document refers to the same terminology used in [I-D.ietf-clue-framework] and in [RFC7262]. We briefly recall herein some of the main terms used in the document. The definition of "CLUE Participant" herein proposed is not imported from any of the above documents.

CLUE Participant: An entity able to use the CLUE protocol within a telepresence session. It can be an endpoint or an MCU able to use the CLUE protocol.

CLUE-capable device: A device that supports the CLUE data channel [I-D.ietf-clue-datachannel], the CLUE protocol and the principles of CLUE negotiation, and seeks CLUE-enabled calls.

Endpoint: The logical point of final termination through receiving, decoding and rendering, and/or initiation through capturing, encoding, and sending of media streams. An endpoint consists of one or more physical devices which source and sink media streams, and exactly one [RFC4353] Participant (which, in turn, includes

exactly one SIP User Agent). Endpoints can be anything from multiscreen/multicamera room controllers to handheld devices.

MCU: Multipoint Control Unit (MCU) - a device that connects two or more endpoints together into one single multimedia conference [RFC5117]. An MCU may include a Mixer [RFC4353].

Media: Any data that, after suitable encoding, can be conveyed over RTP, including audio, video or timed text.

Media Capture: A "Media Capture", or simply "Capture", is a source of Media.

Capture Encoding: A specific encoding of a Media Capture, to be sent via RTP [RFC3550].

Media Stream: The term "Media Stream", or simply "Stream", is used as a synonymous of Capture Encoding.

Media Provider: A CLUE Participant (i.e., an Endpoint or an MCU) able to send Media Streams.

Media Consumer: A CLUE Participant (i.e., an Endpoint or an MCU) able to receive Media Streams.

3. Overview of the CLUE protocol

The CLUE protocol is conceived to enable CLUE telepresence sessions. It is designed in order to address SDP limitations in terms of the description of some information about the multimedia streams that are involved in a real-time multimedia conference. Indeed, by simply using SDP we are not able to convey information about the features of the flowing multimedia streams that are needed to enable a "being there" rendering experience. Such information is designed in the CLUE framework document and formally defined and described in the CLUE data model document. The CLUE protocol represents the mechanism for the exchange of CLUE information between CLUE Participants. It mainly provides the messages to enable a Media Provider to advertise its telepresence capabilities and to enable a Media Consumer to select the desired telepresence options.

The CLUE protocol, as defined in the following, is a stateful, client-server, XML-based application protocol. CLUE protocol messages flow on a reliable and ordered SCTP over DTLS transport channel connecting two CLUE Participants. Messages carry information taken from the XML-based CLUE data model ([I-D.ietf-clue-data-model-schema]). Three main communication layers can be identified:

1. Establishment of the CLUE data channel: in this phase, the CLUE data channel setup takes place. If it completes successfully, the CPs are able to communicate and start the initiation phase.
2. Negotiation of the CLUE protocol version and options (initiation phase): the CPs connected via the CLUE data channel agree on the version and on the options to be used during the telepresence session. Special CLUE messages are used for such a task (OPTIONS and OPTIONS RESPONSE). The version and options negotiation can be performed once and only at this stage. At the end of that basic negotiation, each CP starts its activity as a CLUE MP and/or CLUE MC.
3. CLUE telepresence capabilities description and negotiation: in this phase, the MP-MC dialogues take place on the data channel by means of the CLUE protocol messages.

As soon as the channel is ready, the CLUE Participants must agree on the protocol version and extensions to be used within the telepresence session. CLUE protocol version numbers are characterized by a major version number and a minor version number, both unsigned integers, separated by a dot. While minor version numbers denote backward compatible changes in the context of a given major version, different major version numbers generally indicate a lack of interoperability between the protocol implementations. In order to correctly establish a CLUE dialogue, the involved CPs MUST have in common a major version number (see Section 7 for further details). The subset of the protocol options and extensions that are allowed within the CLUE session is also determined in the initiation phase, such subset being the one including only the options that are supported by both parties. A mechanism for the negotiation of the CLUE protocol version and extensions is envisioned in the initiation phase. According to such a solution, the CP which is the CLUE Channel initiator (CI) issues a proper CLUE message (OPTIONS) to the CP which is the Channel Receiver (CR) specifying the supported version and extensions. The CR then answers by selecting the subset of the CI extensions that it is able to support and determines the protocol version to be used.

After that negotiation phase is completed, CLUE Participants describe and agree on the media flows to be exchanged. Indeed, being CPs A and B both transmitting and receiving, it is possible to distinguish between two dialogues:

1. the one needed to describe and set up the media streams sent from A to B, i.e., the dialogue between A's Media Provider side and B's Media Consumer side

2. the one needed to describe and set up the media streams sent from B to A, i.e., the dialogue between B's Media Provider side and A's Media Consumer side

CLUE messages for the media session description and negotiation are designed by considering the MP side as the server side of the protocol, since it produces and provides media streams, and the MC side as the client side of the protocol, since it requests and receives media streams. The messages that are exchanged to set up the telepresence media session are described by focusing on a single MP-MC dialogue.

The MP first advertises its available media captures and encoding capabilities to the MC, as well as its simultaneity constraints, according to the information model defined in [I-D.ietf-clue-framework]. The CLUE message conveying the MP's multimedia offer is the ADVERTISEMENT message. Such message leverages the XML data model definitions provided in [I-D.ietf-clue-data-model-schema].

The MC selects the desired streams of the MP by using the CONFIGURE message, which makes reference to the information carried in the previously received ADVERTISEMENT.

Besides ADVERTISEMENT and CONFIGURE, other messages have been conceived in order to provide all the needed mechanisms and operations. Such messages will be detailed in the following sections.

4. Protocol messages

CLUE protocol messages are textual, XML-based messages that enable the configuration of the telepresence session. The formal definition of such messages is provided in the XML Schema provided at the end of this document (Section 9).

The XML definitions of the CLUE information provided in [I-D.ietf-clue-data-model-schema] are included within some CLUE protocol messages (namely the ADVERTISEMENT and the CONFIGURE messages), in order to use the concepts defined in [I-D.ietf-clue-framework].

The CLUE protocol messages are the following:

- o OPTIONS
- o OPTIONS RESPONSE

- o ADVERTISEMENT (ADV)
- o ADVERTISEMENT ACKNOWLEDGEMENT (ACK)
- o CONFIGURE (CONF)
- o CONFIGURE RESPONSE (CONF RESPONSE)

While the OPTIONS and OPTIONS RESPONSE messages are exchanged in the initiation phase between the CPs, the other messages are involved in MP-MC dialogues.

Each CLUE message inherits a basic structure depicted in the following excerpt:

```
<!-- CLUE MESSAGE TYPE -->
<xs:complexType name="clueMessageType" abstract="true">
  <xs:sequence>
    <xs:element name="clueId" type="xs:string"/>
    <xs:element name="sequenceNr" type="xs:unsignedInt"/>
  </xs:sequence>
  <xs:attribute name="protocol" type="xs:string" fixed="CLUE" use="required"/>
  <xs:attribute name="v" type="versionType" use="required"/>
</xs:complexType>

<!-- VERSION TYPE -->
<xs:simpleType name="versionType">
  <xs:restriction base="xs:string">
    <xs:pattern value="([0-9])+\.([0-9]+)"></xs:pattern>
  </xs:restriction>
</xs:simpleType>
```

The basic structure determines the mandatory information that is carried within each CLUE message. Such an information is made by:

- o clueId: an XML element containing the identifier of the CP within the telepresence system;
- o sequenceNr: an XML element containing the local message sequence number;
- o protocol: a mandatory attribute set to "CLUE", identifying the protocol the messages refer to;

- o v: a mandatory attribute carrying the version of the protocol. The content of the "v" attribute is composed by the major version number followed by a dot and then by the minor version number of the CLUE protocol in use. Allowed values are of this kind: "1.3", "2.45", etc.

Each CP should manage up to three streams of sequence numbers: (i) one for the messages exchanged in the initiation phase, (ii) one for the messages exchanged as MP, and (iii) one for the messages exchanged as MC.

4.1. OPTIONS

The OPTIONS message is sent by the CP which is the CI to the CP which is the CR as soon as the CLUE data channel is ready. Besides the information envisioned in the basic structure, it specifies:

- o mediaProvider: a mandatory boolean field set to "true" if the CP is able to act as a MP
- o mediaConsumer: a mandatory boolean field set to "true" if the CP is able to act as a MC
- o supportedVersions: the list of the supported versions
- o supportedOptions: the list of the supported options

The XML Schema of such a message is reported below:

```
<!-- CLUE OPTIONS -->
<xs:complexType name="optionsMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <xs:element name="mediaProvider" type="xs:boolean"/>
        <xs:element name="mediaConsumer" type="xs:boolean"/>
        <xs:element name="supportedVersions" type="versionsListType" minOccurs="0"/>
        <xs:element name="supportedOptions" type="optionsListType" minOccurs="0"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- VERSIONS LIST TYPE -->
<xs:complexType name="versionsListType">
  <xs:sequence>
    <xs:element name="version" type="versionType" minOccurs="1"
      maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- OPTIONS LIST TYPE -->
<xs:complexType name="optionsListType">
  <xs:sequence>
    <xs:element name="option" type="optionType" minOccurs="1"
      maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- OPTION TYPE -->
<xs:complexType name="optionType">
  <xs:sequence>
    <xs:element name="name" type="xs:string" />
    <xs:element name="schemaRef" type="xs:anyURI" minOccurs="0"/>
    <xs:element name="version" type="versionType" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

<supportedVersions> contains the list of the versions that are supported by the CI, each one represented in a child <version> element. The content of each <version> element is a string made by the major version number followed by a dot and then by the minor version number (e.g., 1.3 or 2.43). Only one <version> element SHOULD be provided for each major version supported, containing the maximum minor version number of such a version, since all minor versions are backward compatible. If no <supportedVersions> is carried within the OPTIONS message, the CI supports only the version declared in the "v" attribute and all the versions having the same major version number and lower minor version number. For example, if the "v" attribute has a value of "3.4" and there is no <supportedVersions> tag in the OPTIONS message, it means the CI supports only major version 3 with all the minor versions comprised between 3.0 and 3.4, with version 3.4 included. If a <supportedVersion> is provided, at least one <version> tag MUST be included.

The <supportedOptions> element specifies the list of options supported by the CI. If there is no <supportedOptions> in the OPTIONS message, the CI does not support anything other than what is envisioned in the versions it supports. For each option, an <option> element is provided. An option is characterized by a name, an XML schema of reference where the option is defined, and the version of the protocol which the option refers to.

4.2. OPTIONS RESPONSE

The OPTIONS RESPONSE is sent by a CR to a CI as a reply to the OPTIONS message. As depicted in the figure below, the OPTIONS RESPONSE contains mandatorily a response code and a response string indicating the processing result of the OPTIONS message. Following, the CR attaches two boolean tags, <mediaProvider> and <mediaConsumer>, expressing the supported roles in terms of respectively MP and MC, similarly to what the CI does in the OPTIONS message. These two elements are optional in the OPTIONS RESPONSE since, in case of error response code, the CR might not want to add further information besides the response code and the reason string. In case of no errors, the CR MUST insert within the OPTIONS RESPONSE the <mediaProvider> and the <mediaConsumer> elements. Finally, the highest commonly supported version number is expressed in the <version> field. The content of the <version> element MUST be a string made by the major version number followed by a dot and then by the minor version number (e.g., 1.3 or 2.43). The commonly supported options are copied in the the <commonOptions> field.


```
<!-- CLUE OPTIONS RESPONSE -->
<xs:complexType name="optionsResponseMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <xs:element name="responseCode" type="xs:short"/>
        <xs:element name="reasonString" type="xs:string"/>
        <xs:element name="mediaProvider" type="xs:boolean" minOccurs="0"/>
        <xs:element name="mediaConsumer" type="xs:boolean" minOccurs="0"/>
        <xs:element name="version" type="versionType" minOccurs="0"/>
        <xs:element name="commonOptions" type="optionsListType" minOccurs="0"/>
        <xs:any namespace="##other"
          processContents="lax" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

After the reception of such a message, the version to be used is determined by each part of the conversation. Indeed, it is the one provided in the <version> tag of the OPTIONS RESPONSE message. The following CLUE messages MUST use such a version number in the "v" attribute. The allowed options in the CLUE dialogue will be those indicated in the <commonOptions> of the OPTIONS RESPONSE message.

4.3. ADVERTISEMENT

The ADVERTISEMENT message (ADV) is used by the MP to advertise the available media captures and related information to the MC. The MP sends to the MC an ADV as soon as it is ready after the successful completion of the initiation phase, i.e., as soon as the version and the options of the CLUE protocol are agreed between the CPs. During the telepresence session, the ADV can be sent from the MP both periodically and on a per-event basis, i.e., each time there are changes in the MP's CLUE telepresence capabilities.

The ADV structure is defined in the picture below. The ADV contains elements compliant with the CLUE data model that characterize the MP's telepresence offer. Namely, such elements are: the list of the media captures (<mediaCaptures>), of the encoding groups (<encodingGroups>), of the capture scenes (<captureScenes>) of the global views (<globalViews>), and of the represented participants (<people>). Each of them is fully described in the CLUE framework document and formally defined in the CLUE data model document.

```
<!-- CLUE ADVERTISEMENT MESSAGE TYPE -->
<xs:complexType name="advertisementMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <!-- mandatory -->
        <xs:element name="mediaCaptures" type="dm:mediaCapturesType"/>
        <xs:element name="encodingGroups" type="dm:encodingGroupsType"/>
        <xs:element name="captureScenes" type="dm:captureScenesType"/>
        <!-- optional -->
        <xs:element name="simultaneousSets" type="dm:simultaneousSetsType"
          minOccurs="0"/>
        <xs:element name="globalViews" type="dm:globalViewsType"
          minOccurs="0"/>
        <xs:element name="people" type="dm:peopleType" minOccurs="0"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

4.4. ADVERTISEMENT ACKNOWLEDGEMENT

The ADVERTISEMENT ACKNOWLEDGEMENT message (ACK) is sent by a MC to a MP to acknowledge an ADV message. As it can be seen from the message schema provided in the following, the ACK contains a response code and a reason string for describing the processing result of the ADV. The <advSequenceNr> carries the sequence number of the ADV the ACK refers to.

```
<!-- ADV ACK MESSAGE TYPE -->
<xs:complexType name="advAcknowledgementMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <xs:element name="responseCode" type="xs:short"/>
        <xs:element name="reasonString" type="xs:string"/>
        <xs:element name="advSequenceNr" type="xs:unsignedInt"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

4.5. CONFIGURE

The CONFIGURE message (CONF) is sent from a MC to a MP to list the advertised captures the MC wants to receive. The MC can send a CONF after the reception of an ADV or each time it wants to request other captures that have been previously advertised by the MP. The content of the CONF message is shown below.

```
<!-- CLUE CONFIGURE MESSAGE TYPE -->
<xs:complexType name="configureMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <!-- mandatory fields -->
        <xs:element name="advSequenceNr" type="xs:unsignedInt"/>
        <xs:element name="ack" type="xs:boolean" minOccurs="0" fixed="true"/>
        <xs:element name="captureEncodings" type="dm:captureEncodingsType"
          minOccurs="0"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The <advSequenceNr> element contains the sequence number of the ADV message the CONF refers to.

The optional boolean <ack> element, set to "true", if present, indicates that the CONF message also acknowledge the referred advertisement, by applying in that way a piggybacking mechanism for simultaneously acknowledging and replying to the ADV message. The <ack> element SHOULD not be present at all if an ACK message has been already sent back to the MP.

The most important content of the CONFIGURE message is the list of the capture encodings provided in the <captureEncodings> element. Such an element contains a sequence of capture encodings, representing the streams to be instantiated.

4.6. CONFIGURE RESPONSE

```
<!-- CONFIGURE RESPONSE MESSAGE TYPE -->
<xs:complexType name="configureResponseMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <xs:element name="responseCode" type="xs:short"/>
        <xs:element name="reasonString" type="xs:string"/>
        <xs:element name="confSequenceNr" type="xs:unsignedInt"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The CONFIGURE RESPONSE message (CONF RESPONSE) is sent from the MP to the MC to communicate the processing result of requests carried in the previously received CONF message. It contains a response code with a reason string indicating either the success or the failure (along with failure details) of a CONF request processing. Following, the <confSequenceNr> field contains the sequence number of the CONF message the response refers to.

4.7. Response codes and reason strings

The response codes and strings defined for use with CLUE are as follows:

Response code	Response string	Description
200	Success	The request has been successfully processed.
300	Bad syntax	The XML syntax of the message is not correct.

301	Invalid value	The message contains an invalid parameter value.
302	Conflicting values	The message contains values that cannot be used together.
400	Version not supported	The protocol version used in the message is not supported.
401	Invalid sequencing	The sequence number of the message is out of date.
402	Invalid identifier	The identifier used in the message is not valid or unknown.
403	ADV Expired	The number of the ADV the CONF refers to is out of date.
404	Subset choice not allowed	The subset choice is not allowed for the specified MCC

Response codes are defined as a sequence of three digits. A semantic is associated to the first digit. Response codes that do not begin with "2" or "1" indicate an error response, i.e., that an error occurred while processing a CLUE request. Response codes beginning with "3" indicate problems with the XML content of the message ("Bad syntax", "Invalid value", etc.), while response codes beginning with "4" refer to problems related to CLUE protocol semantics ("Invalid sequencing", "Version not supported", etc.). Response codes beginning with "2" are associated to successful responses. Response codes beginning with "1" could be associated to temporary responses.

Further response codes can be designed in future version of the protocol, provided they do not overwrite the ones here defined and they respect the semantic of the first code digit.

5. Protocol state machines

The CLUE protocol is an application protocol used between two CPs in order to properly configure a multimedia telepresence session. CLUE protocol messages flow upon the CLUE Data Channel, a DTLS/SCTP channel established as depicted in [I-D.ietf-clue-signaling]. Over such a channel there are typically two CLUE streams between the channel terminations flowing in opposite directions. In other words, typically, both channel terminations act simultaneously as a MP and as a MC. We herein discuss the state machines associated, respectively, with the CLUE Participant, with the MC process and with the MP process. Only the CLUE application logic is considered. The interaction of CLUE protocol and SDP negotiations for the media streams to be exchanged is treated in [I-D.ietf-clue-signaling], Section 5.

6. CLUE Participant's state machine

The main state machines focus on the behavior of the CLUE Participant (CP) acting as a CLUE channel initiator/receiver (CI/CR).

The initial state is the IDLE one. When in the IDLE state, the CLUE data channel is not established and no CLUE-controlled media are exchanged between the two considered CLUE-capable devices (if there is an ongoing exchange of media streams, such media streams are not currently CLUE-controlled).

When the CLUE data channel set up starts ("start channel"), the CP moves from the IDLE state to the CHANNEL SETUP state.

If the CLUE data channel is successfully set up ("channel established"), the CP moves from the CHANNEL SETUP state to the OPTIONS state. Otherwise ("channel error"), it moves back to the IDLE state. The same transition happens if the CLUE-enabled telepresence session ends ("session ends"), i.e., when an SDP negotiation for removing the CLUE channel is performed.

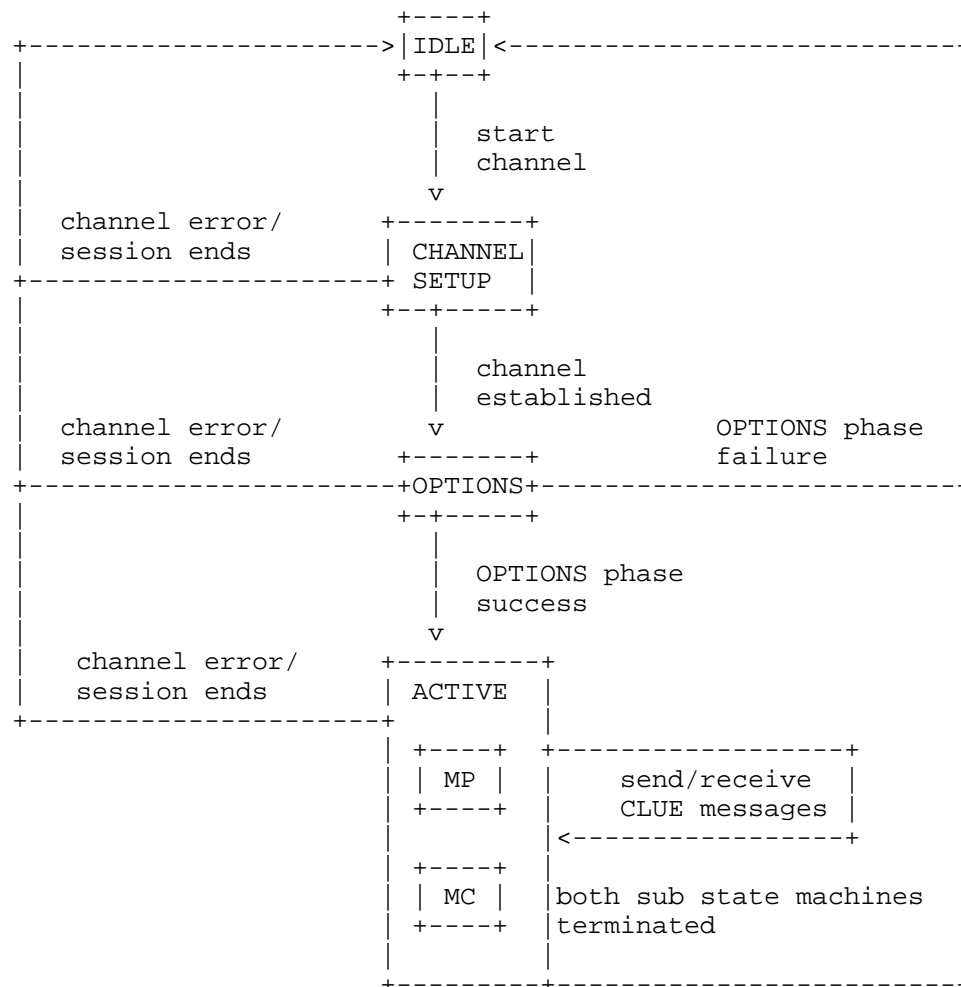
When in the OPTIONS state, the CP addresses the initiation phase where both parts agree on the version and on the options to be used in the subsequent CLUE messages exchange phase. If the CP is the Channel Initiator (CI), it sends an OPTIONS message and waits for the OPTIONS RESPONSE message. If the CP is the Channel Receiver (CR), it waits for the OPTIONS message and, as soon as it arrives, replies with the OPTIONS RESPONSE message. If the negotiation is

successfully completed ("OPTIONS phase success"), the CP moves from the OPTIONS state to the ACTIVE state. If the initiation phase fails ("OPTIONS phase failure"), the CP moves from the OPTIONS state to the IDLE state. The initiation phase might fail because of one of the following reasons:

1. the CI receives an OPTIONS RESPONSE with an error response code
2. the CI does not receive any OPTIONS RESPONSE and a timeout error is raised
3. the CR does not receive any OPTIONS and a timeout error is raised

When in the ACTIVE state, the CP starts the envisioned sub-state machines (i.e., the MP state machine and the MC state machine) according to the roles it plays in the telepresence sessions. Such roles have been previously declared in the OPTIONS and OPTIONS RESPONSE messages involved in the initiation phase (see OPTIONS sections Section 4.1 and Section 4.2 for the details). When in the ACTIVE state, the CP delegates the sending and the processing of the CLUE messages to the appropriate MP/MC sub-state machines. If the CP receives a further OPTIONS/OPTIONS RESPONSE message, it MUST ignore the message and stay in the ACTIVE state.

The CP moves from the ACTIVE state to the IDLE one when the sub-state machines that have been activated are (both) in the relative TERMINATED state (see sections Section 6.1 and Section 6.2).



6.1. Media Provider's state machine

As soon as the sub-state machine of the MP is activated, it is in the ADV state. In the ADV state, the MP is preparing the ADV message reflecting its actual telepresence capabilities.

After the ADV has been sent ("ADV sent"), the MP moves from the ADV state to the WAIT FOR ACK state. If an ACK message with a successful response code arrives ("ACK received"), the MP moves to the WAIT FOR CONF state. If a NACK arrives (i.e., an ACK message with an error

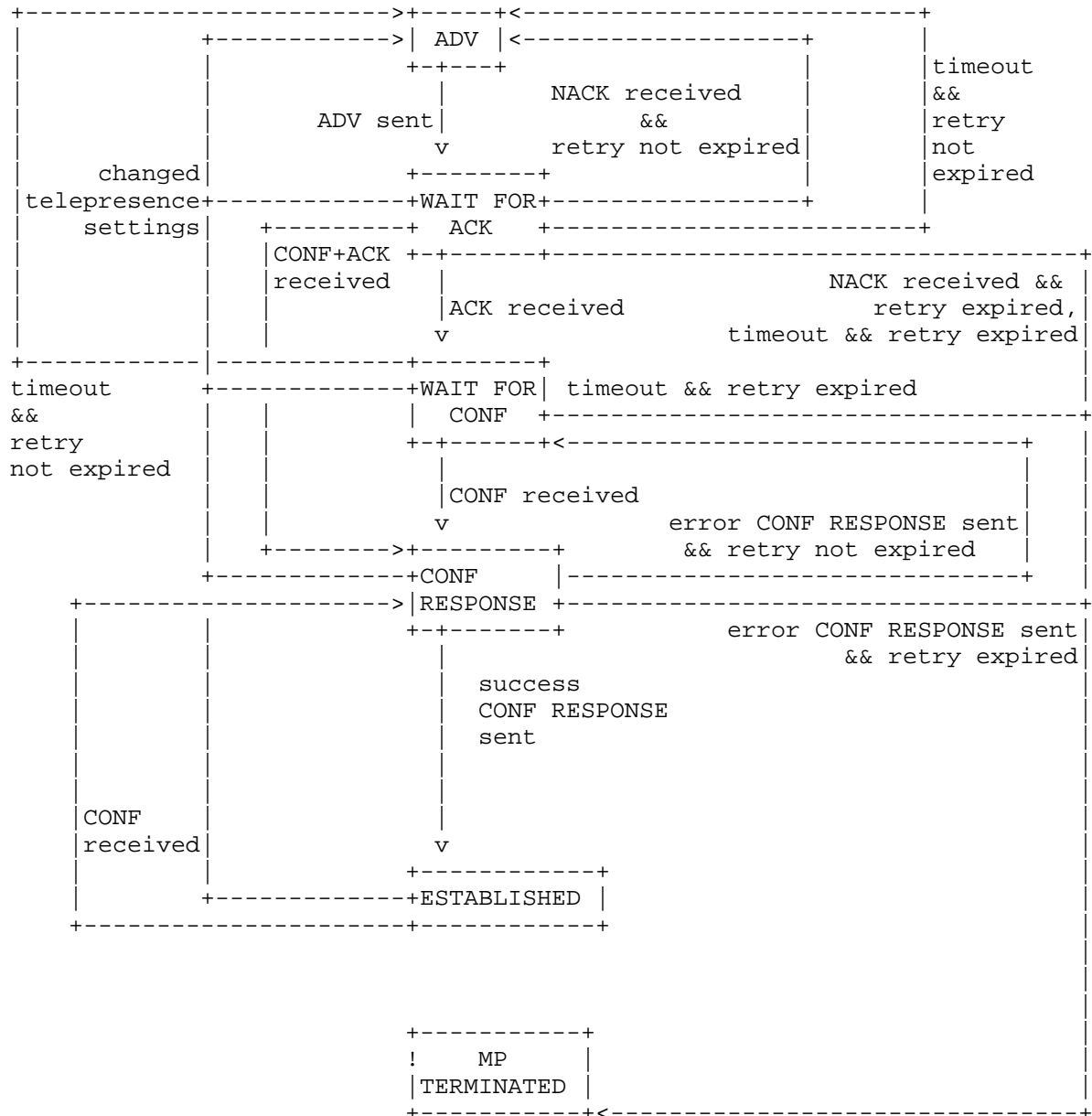
response code), and the number of NACKs for the issued ADV is under the retry threshold ("NACK received && retry not expired"), the MP moves back to the ADV state for preparing a new ADV. If a NACK arrives and the number of received NACKs for that ADV overcome the threshold ("NACK received and && retry expired"), the MP goes to the MP-TERMINATED state. Similarly, if the waiting time for the ACK is fired a number of times under the retry threshold ("timeout && retry not expired"), the MP goes back to the ADV state to send a new copy of the ADV. If the number of retries overcomes the threshold ("timeout && retry expired"), the MP moves from the WAIT FOR ACK state to the MP-TERMINATED state. When in the WAIT FOR ACK state, if a CONF+ACK message arrives ("CONF+ACK received"), the MP goes directly to the CONF RESPONSE state. CONF+ACK messages referring to out-of-date ADVs MUST be ignored, i.e., they do not trigger any state transition. If the telepresence settings of the MP change while in the WAIT FOR ACK state ("changed telepresence settings"), the MP switches from the WAIT FOR ACK state to the ADV state to create a new ADV.

When in the WAIT FOR CONF state, the MP listens to the channel for a CONF request coming from the MC. If a CONF arrives ("CONF received"), the MP switches to the CONF RESPONSE state. If the CONF does not arrive within the timeout interval and the retry threshold has not been overcome ("timeout && retry not expired"), the MP moves back to the ADV state. When the retry expires ("timeout && retry expired") the MP moves to the MP TERMINATED state. If the telepresence settings change in the meanwhile ("changed telepresence settings"), the MP moves from the WAIT FOR CONF back to the ADV state to create the new ADV to be sent to the MC.

The MP in the CONF RESPONSE state processes the received CONF in order to produce a CONF RESPONSE message. If the MP is fine with the MC's configuration, then it sends a 200 CONF RESPONSE ("success CONF RESPONSE sent") and moves to the ESTABLISHED state. If there are errors in the CONF processing, then the MP issues a CONF RESPONSE carrying an error response code and, if under the retry threshold ("error CONF RESPONSE sent && retry not expired"), it goes back to the WAIT FOR CONF state to wait for a new configuration request. If the number of trials exceeds the retry threshold ("error CONF RESPONSE sent && retry expired"), the state MP TERMINATED is reached. Finally, if there are changes in the MP's telepresence settings ("changed telepresence settings"), the MP switches to the ADV state.

The MP in the ESTABLISHED state has successfully negotiated the media streams with the MC by means of the CLUE messages. If there are changes in the MP's telepresence settings ("changed telepresence settings"), the MP moves back to the ADV state. In the ESTABLISHED state, the CLUE-controlled media streams of the session are those

described in the last successfully processed CONF message.



6.2. Media Consumer's state machine

As soon as the sub-state machine of the MC is activated, it is in the WAIT FOR ADV state. An MC in the WAIT FOR ADV state is waiting for an ADV coming from the MP. If the ADV arrives ("ADV received"), the MC reaches the ADV PROCESSING state. Otherwise, the MC is stuck in the WAIT FOR ADV state.

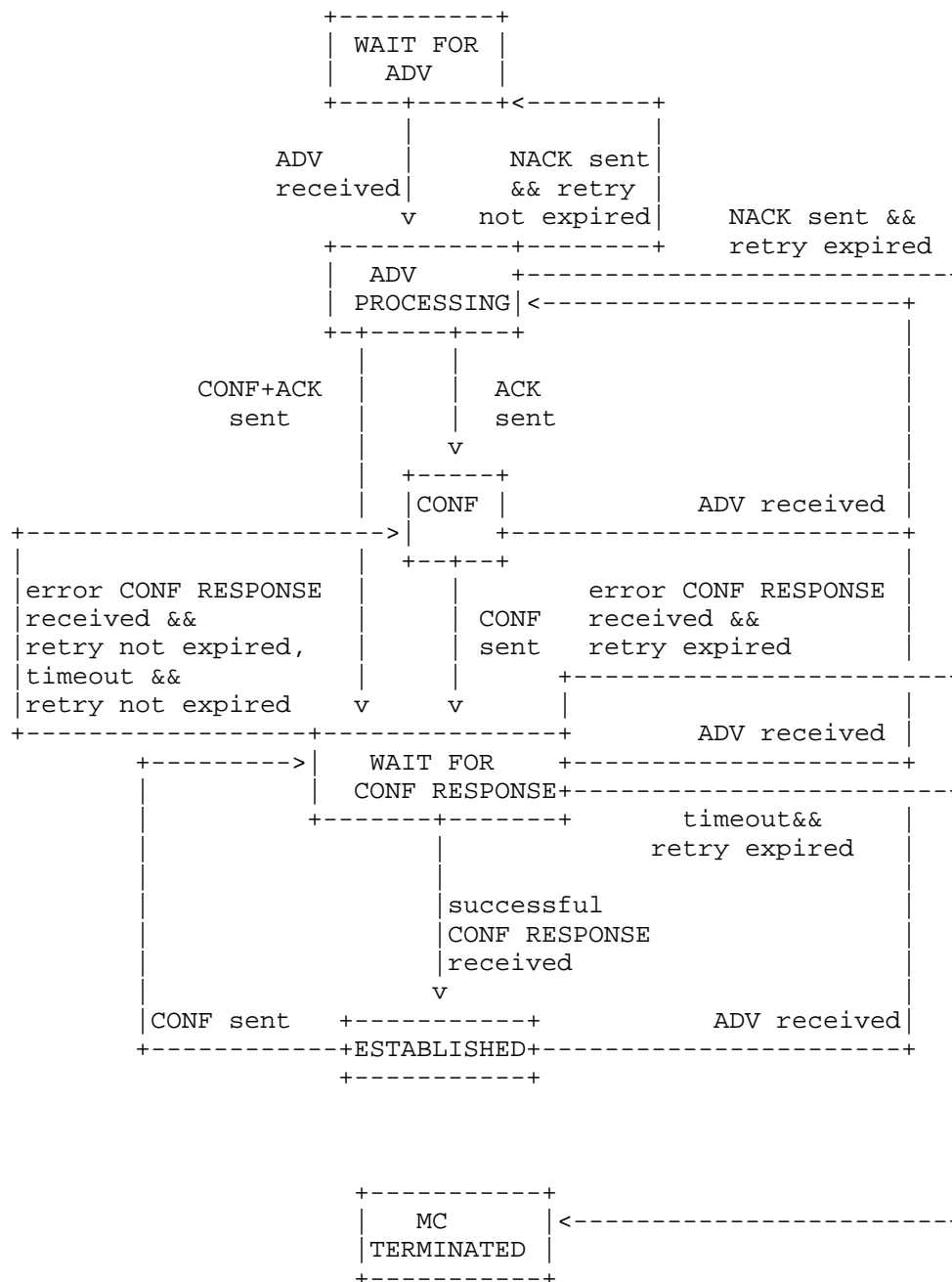
In the ADV PROCESSING state, the ADV is parsed by the MC. If the ADV is successfully processed, there are two possibilities. According to the first one, the MC issues a successful ACK message to the MP ("ACK sent") and moves to the CONF state. In the second one, the MC prepares and sends a CONF message with the <ack> field set to "true" ("CONF+ACK sent") and goes directly to the WAIT FOR CONF RESPONSE state.

If the ADV elaboration is unsuccessful (bad syntax, missing XML elements, etc.), and the number of times this condition has been verified is under the retry threshold, the MC sends a NACK message (i.e., an ACK with an error response code) to the MP describing the problem via a proper reason phrase. By this way ("NACK sent && retry not expired"), the MC switches back to the WAIT FOR ADV state, waiting for a new ADV. If the NACK retry expires ("NACK sent && retry expired"), the MC moves to the MC TERMINATED state.

When in the CONF state, the MC is preparing the CONF request to be issued to the MP on the basis of the previously ACK-ed ADV. When the CONF has been sent ("CONF sent"), the MC moves to the WAIT FOR CONF RESPONSE state. If a new ADV arrives in the meanwhile ("ADV received"), the MC goes back to the ADV PROCESSING state.

In the WAIT FOR CONF RESPONSE state, the MC is waiting for the MP's response to the issued CONF or for the issued CONF+ACK. If a 200 CONF RESPONSE message is received ("successful CONF RESPONSE received"), it means that the MP and the MC have successfully agreed on the media streams to be shared. Then, the MC can move to the ESTABLISHED state. On the other hand, if an error response is received and the associated retry counter does not overcome the threshold ("error CONF RESPONSE received && retry not expired"), the MC moves back to the CONF state to prepare a new CONF request. In case of "error CONF RESPONSE received && retry expired", the MC moves to the MC TERMINATED state. If no CONF RESPONSE arrives and the number of timeouts is under the threshold ("timeout && retry not expired"), the MC moves to the CONF state and sends again the CONF message. If no CONF RESPONSE arrives and the number of timeouts is over the threshold ("timeout && retry expired"), the MC moves to the MC TERMINATED state. If a new ADV is received in the WAIT FOR CONF RESPONSE state, the MC switches to the ADV PROCESSING state.

When the MC is in the ESTABLISHED state, the telepresence session configuration has been set up at the CLUE application level according to the MC's preferences. Both the MP and the MC have agreed on (and are aware of) the CLUE-controlled media streams to be exchanged within the call. While in the ESTABLISHED state, it might happen that the MC decides to change something in the call settings. The MC then issues a new CONF ("CONF sent") and goes to wait for the new CONF RESPONSE in the WAIT FOR CONF RESPONSE state. On the other hand, in the ESTABLISHED state, if a new ADV arrives from the MP ("ADV received"), it means that something has changed on the MP's side. The MC then moves to the ADV PROCESSING state.



7. Versioning

CLUE protocol messages are XML messages compliant to the CLUE protocol XML schema. The version of the protocol corresponds to the version of the schema. Both client and server have to test the compliance of the received messages with the XML schema of the CLUE protocol. If the compliance is not verified, the message cannot be processed further.

Obviously, client and server cannot communicate if they do not share exactly the same XML schema. Such a schema is the one included in the yet to come RFC, and associated with the CLUE URN "urn:ietf:params:xml:ns:clue-protocol". If all CLUE-enabled devices use that schema there will be no interoperability problems due to schema issues.

The version of the XML schema contained in the standard document deriving from this draft will be 1.0. The version usage is similar in philosophy to XMPP ([RFC6120]). A version number has major and minor components, each a non-negative integer. Major version changes denote non-interoperable changes. Minor version changes denote schema changes that are backward compatible by ignoring unknown XML elements, or other backward compatible changes.

The minor versions of the XML schema MUST be backward compatible, not only in terms of schema but also semantically and procedurally as well. This means that they should define further features and functionality besides those defined in the previous versions, in an incremental way, without impacting the basic rules defined in the previous version of the schema. In this way, if a MP is able to speak, e.g., version 1.5 of the protocol while the MC only understands version 1.4, the MP should have no problem in reverting the dialogue back to version 1.4 without exploiting 1.5 features and functionality.

It is expected that, before the CLUE protocol XML schema reaches a steady state, prototypes developed by different organizations will conduct interoperability testing. In that case, in order to interoperate, they have to be compliant to the current version of the XML schema, i.e., the one copied in the most up-to-date version of the draft defining the CLUE protocol. The versions of the non-standard XML schema will be numbered as 0.01, 0.02, and so on. During the standard development phase, the versions of the XML schema will probably not be backward compatible so it is left to prototype implementers the responsibility of keeping their products up to date.

8. Extensions and options

Although the standard version of the CLUE protocol XML schema is designed to thoroughly cope with the requirements emerging from the application domain, new needs might arise and extensions can be designed. Extensions specify information and behaviors that are not described in a certain version of the protocol. They can relate to:

1. new information, to be carried in the existing messages. For example, we may want to add more fields within an existing message;
2. new messages. This is the case if there is no proper message for a certain task, so a brand new CLUE message needs to be defined.

As to the first type of extensions, it is possible to distinguish between protocol-specific and data model information. Indeed, CLUE messages are envelopes carrying both:

- o (i) XML elements defined within the CLUE protocol XML schema itself (protocol-specific information)
- o (ii) other XML elements compliant to the CLUE data model schema (data model information)

When new protocol-specific information is needed somewhere in the protocol messages, it can be added in place of the `<any>` elements and `<anyAttribute>` elements envisioned by the protocol schema. The policy currently defined in the protocol schema for handling `<any>` and `<anyAttribute>` elements is:

- o `elementFormDefault="qualified"`
- o `attributeFormDefault="unqualified"`

In that case, the new information must be qualified by namespaces other than `"urn:ietf:params:xml:ns:clue-protocol"` (the protocol URN) and `"urn:ietf:params:xml:ns:clue-info"` (the data model URN). Elements or attributes from unknown namespaces MUST be ignored.

The other matter concerns data model information. Data model information is defined by the XML schema associated with the URN `"urn:ietf:params:xml:ns:clue-info"`. Also for the XML elements defined in such a schema there are extensibility issues. Those issues are overcome by using `<any>` and `<anyAttribute>` placeholders. Similarly to what said before, new information within data model elements can be added in place of `<any>` and `<anyAttribute>` schema elements, as long as they are properly namespace qualified.

On the other hand (second type of extensions), "extra" CLUE protocol messages, i.e., messages not envisioned in the latest standard version of the schema, can be needed. In that case, the messages and the associated behavior should be defined in external documents that both communication parties must be aware of.

Both types of extensions, i.e., new information and new messages, can be characterized by:

- o a name;
- o an external XML Schema defining the XML information and/or the XML messages representing the extension;
- o the standard version of the protocol the extension refers to.

For that reason, the extensions can be represented by means of the <option> element as defined below, which is carried within the OPTIONS and OPTIONS RESPONSE messages to represent the extensions supported by the CI and by the CR.

```
<!-- OPTION TYPE -->
<xs:complexType name="optionType">
  <xs:sequence>
    <xs:element name="name" type="xs:string" />
    <xs:element name="schemaRef" type="xs:anyURI" minOccurs="0"/>
    <xs:element name="version" type="versionType" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

9. XML Schema

In this section, the XML schema defining the CLUE messages is provided.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  version="0.4"
  targetNamespace="urn:ietf:params:xml:ns:clue-protocol"
  xmlns:tns="urn:ietf:params:xml:ns:clue-protocol"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
```



```
xmlns:dm="urn:ietf:params:xml:ns:clue-info"
xmlns="urn:ietf:params:xml:ns:clue-protocol"
elementFormDefault="qualified"
attributeFormDefault="unqualified">

<!-- Import data model schema -->
<xs:import namespace="urn:ietf:params:xml:ns:clue-info"
schemaLocation="data-model-schema-09.xsd"/>

<!-- ELEMENT DEFINITIONS -->
<xs:element name="options" type="optionsMessageType"/>
<xs:element name="optionsResponse" type="optionsResponseMessageType"/>
<xs:element name="advertisement" type="advertisementMessageType"/>
<xs:element name="ack" type="advAcknowledgementMessageType"/>
<xs:element name="configure" type="configureMessageType"/>
<xs:element name="configureResponse" type="configureResponseMessageType"/>

<!-- CLUE MESSAGE TYPE -->
<xs:complexType name="clueMessageType" abstract="true">
<xs:sequence>
<xs:element name="clueId" type="xs:string"/>
<xs:element name="sequenceNr" type="xs:unsignedInt"/>
</xs:sequence>
<xs:attribute name="protocol" type="xs:string" fixed="CLUE" use="required"/>
<xs:attribute name="v" type="versionType" use="required"/>
</xs:complexType>

<!-- VERSION TYPE -->
<xs:simpleType name="versionType">
<xs:restriction base="xs:string">
<xs:pattern value="([0-9])+\.([0-9]+)"></xs:pattern>
</xs:restriction>
</xs:simpleType>

<!-- RESPONSE CODE TYPE -->
<xs:simpleType name="responseCodeType">
<xs:restriction base="xs:integer">
<xs:pattern value="[1-9][0-9][0-9]"/>
</xs:restriction>
</xs:simpleType>

<!-- CLUE OPTIONS -->
<xs:complexType name="optionsMessageType">
<xs:complexContent>
<xs:extension base="clueMessageType">
<xs:sequence>
```

```
<xs:element name="mediaProvider" type="xs:boolean"/>
<xs:element name="mediaConsumer" type="xs:boolean"/>
<xs:element name="supportedVersions" type="versionsListType" minOccurs="0"/>
<xs:element name="supportedOptions" type="optionsListType" minOccurs="0"/>
<xs:any namespace="##other" processContents="lax" minOccurs="0"/>
</xs:sequence>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- VERSIONS LIST TYPE -->
<xs:complexType name="versionsListType">
  <xs:sequence>
    <xs:element name="version" type="versionType" minOccurs="1"
      maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- OPTIONS LIST TYPE -->
<xs:complexType name="optionsListType">
  <xs:sequence>
    <xs:element name="option" type="optionType" minOccurs="1"
      maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- OPTION TYPE -->
<xs:complexType name="optionType">
  <xs:sequence>
    <xs:element name="name" type="xs:string" />
    <xs:element name="schemaRef" type="xs:anyURI" minOccurs="0"/>
    <xs:element name="version" type="versionType" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- CLUE OPTIONS RESPONSE -->
<xs:complexType name="optionsResponseMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <xs:element name="responseCode" type="responseCodeType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
<xs:element name="reasonString" type="xs:string"/>
<xs:element name="mediaProvider" type="xs:boolean" minOccurs="0"/>
<xs:element name="mediaConsumer" type="xs:boolean" minOccurs="0"/>
<xs:element name="version" type="versionType" minOccurs="0"/>
<xs:element name="commonOptions" type="optionsListType" minOccurs="0"/>
<xs:any namespace="##other"
processContents="lax" minOccurs="0"/>
</xs:sequence>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- CLUE ADVERTISEMENT MESSAGE TYPE -->
<xs:complexType name="advertisementMessageType">
<xs:complexContent>
<xs:extension base="clueMessageType">
<xs:sequence>
<!-- mandatory -->
<xs:element name="mediaCaptures" type="dm:mediaCapturesType"/>
<xs:element name="encodingGroups" type="dm:encodingGroupsType"/>
<xs:element name="captureScenes" type="dm:captureScenesType"/>
<!-- optional -->
<xs:element name="simultaneousSets" type="dm:simultaneousSetsType"
minOccurs="0"/>
<xs:element name="globalViews" type="dm:globalViewsType"
minOccurs="0"/>
<xs:element name="people" type="dm:peopleType" minOccurs="0"/>
<xs:any namespace="##other" processContents="lax" minOccurs="0"/>
</xs:sequence>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- ACK MESSAGE TYPE -->
<xs:complexType name="advAcknowledgementMessageType">
<xs:complexContent>
<xs:extension base="clueMessageType">
<xs:sequence>
<xs:element name="responseCode" type="responseCodeType"/>
<xs:element name="reasonString" type="xs:string"/>
<xs:element name="advSequenceNr" type="xs:unsignedInt"/>
<xs:any namespace="##other" processContents="lax" minOccurs="0"/>
</xs:sequence>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:extension>
</xs:complexContent>
```

```
</xs:complexType>

<!-- CLUE CONFIGURE MESSAGE TYPE -->
<xs:complexType name="configureMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <xs:element name="advSequenceNr" type="xs:unsignedInt"/>
        <xs:element name="ack" type="xs:boolean"
          minOccurs="0" fixed="true"/>
        <xs:element name="captureEncodings" type="dm:captureEncodingsType"
          minOccurs="0"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- CONFIGURE RESPONSE MESSAGE TYPE -->
<xs:complexType name="configureResponseMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <xs:element name="responseCode" type="responseCodeType"/>
        <xs:element name="reasonString" type="xs:string"/>
        <xs:element name="confSequenceNr" type="xs:unsignedInt"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

</xs:schema>
```

10. Examples

In the following we provide an example of ADVERTISEMENT representing the telepresence environment described in [I-D.ietf-clue-data-model-schema], Section "Sample XML file" and Section "MCC example" respectively.

10.1. Simple ADV

The associated Media Provider's telepresence capabilities are described in [I-D.ietf-clue-data-model-schema], Section "Sample XML

file".

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<advertisement xmlns="urn:ietf:params:xml:ns:clue-protocol"
  xmlns:ns2="urn:ietf:params:xml:ns:clue-info"
  xmlns:ns3="urn:ietf:params:xml:ns:vcard-4.0"
  protocol="CLUE" v="0.4">
  <clueId>Napoli</clueId>
  <sequenceNr>45</sequenceNr>
  <mediaCaptures>
    <ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="ns2:videoCaptureType" captureID="AC0" mediaType="video">
      <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
      <ns2:encGroupIDREF>EG1</ns2:encGroupIDREF>
      <ns2:spatialInformation>
        <ns2:capturePoint>
          <ns2:x>0.5</ns2:x>
          <ns2:y>1.0</ns2:y>
          <ns2:z>0.5</ns2:z>
          <ns2:lineOfCapturePoint>
            <ns2:x>0.5</ns2:x>
            <ns2:y>0.0</ns2:y>
            <ns2:z>0.5</ns2:z>
          </ns2:lineOfCapturePoint>
        </ns2:capturePoint>
      </ns2:spatialInformation>
      <ns2:individual>true</ns2:individual>
      <ns2:description lang="en">main audio from the room</ns2:description
    >
      <ns2:priority>1</ns2:priority>
      <ns2:lang>it</ns2:lang>
      <ns2:mobility>static</ns2:mobility>
      <ns2:view>room</ns2:view>
      <ns2:capturedPeople>
        <ns2:personIDREF>alice</ns2:personIDREF>
        <ns2:personIDREF>bob</ns2:personIDREF>
        <ns2:personIDREF>ciccio</ns2:personIDREF>
      </ns2:capturedPeople>
    </ns2:mediaCapture>
    <ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="ns2:videoCaptureType" mediaType="video" captureID="VC0">
      <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
      <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
      <ns2:spatialInformation>
        <ns2:capturePoint>
          <ns2:x>0.5</ns2:x>
          <ns2:y>1.0</ns2:y>
```

```
<ns2:z>0.5</ns2:z>
<ns2:lineOfCapturePoint>
  <ns2:x>0.5</ns2:x>
  <ns2:y>0.0</ns2:y>
  <ns2:z>0.5</ns2:z>
</ns2:lineOfCapturePoint>
</ns2:capturePoint>
</ns2:spatialInformation>
<ns2:individual>true</ns2:individual>
<ns2:description lang="en">left camera video capture
</ns2:description>
<ns2:priority>1</ns2:priority>
<ns2:lang>it</ns2:lang>
<ns2:mobility>static</ns2:mobility>
<ns2:view>individual</ns2:view>
<ns2:capturedPeople>
  <ns2:personIDREF>ciccio</ns2:personIDREF>
</ns2:capturedPeople>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" mediaType="video" captureID="VC1">
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
  <ns2:spatialInformation>
    <ns2:capturePoint>
      <ns2:x>0.5</ns2:x>
      <ns2:y>1.0</ns2:y>
      <ns2:z>0.5</ns2:z>
      <ns2:lineOfCapturePoint>
        <ns2:x>0.5</ns2:x>
        <ns2:y>0.0</ns2:y>
        <ns2:z>0.5</ns2:z>
      </ns2:lineOfCapturePoint>
    </ns2:capturePoint>
  </ns2:spatialInformation>
  <ns2:individual>true</ns2:individual>
  <ns2:description lang="en">central camera video capture
  </ns2:description>
  <ns2:priority>1</ns2:priority>
  <ns2:lang>it</ns2:lang>
  <ns2:mobility>static</ns2:mobility>
  <ns2:view>individual</ns2:view>
  <ns2:capturedPeople>
    <ns2:personIDREF>alice</ns2:personIDREF>
  </ns2:capturedPeople>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" mediaType="video" captureID="VC2">
```

```
<ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
<ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
<ns2:spatialInformation>
  <ns2:capturePoint>
    <ns2:x>0.5</ns2:x>
    <ns2:y>1.0</ns2:y>
    <ns2:z>0.5</ns2:z>
    <ns2:lineOfCapturePoint>
      <ns2:x>0.5</ns2:x>
      <ns2:y>0.0</ns2:y>
      <ns2:z>0.5</ns2:z>
    </ns2:lineOfCapturePoint>
  </ns2:capturePoint>
</ns2:spatialInformation>
<ns2:individual>true</ns2:individual>
<ns2:description lang="en">right camera video capture
</ns2:description>
<ns2:priority>1</ns2:priority>
<ns2:lang>it</ns2:lang>
<ns2:mobility>static</ns2:mobility>
<ns2:view>individual</ns2:view>
<ns2:capturedPeople>
  <ns2:personIDREF>bob</ns2:personIDREF>
</ns2:capturedPeople>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" mediaType="video" captureID="VC3">
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
  <ns2:nonSpatiallyDefinable>true</ns2:nonSpatiallyDefinable>
  <ns2:composed>false</ns2:composed>
  <ns2:switched>true</ns2:switched>
  <ns2:policy>Soundlevel:0</ns2:policy>
  <ns2:maxCaptures>1</ns2:maxCaptures>
  <ns2:description lang="en">loudest room segment</ns2:description>
  <ns2:priority>1</ns2:priority>
  <ns2:lang>it</ns2:lang>
  <ns2:mobility>static</ns2:mobility>
  <ns2:view>individual</ns2:view>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" mediaType="video" captureID="VC4">
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
  <ns2:spatialInformation>
    <ns2:capturePoint>
      <ns2:x>0.5</ns2:x>
      <ns2:y>1.0</ns2:y>
```

```
<ns2:z>0.5</ns2:z>
<ns2:lineOfCapturePoint>
  <ns2:x>0.5</ns2:x>
  <ns2:y>0.0</ns2:y>
  <ns2:z>0.5</ns2:z>
</ns2:lineOfCapturePoint>
</ns2:capturePoint>
</ns2:spatialInformation>
<ns2:individual>true</ns2:individual>
<ns2:description lang="en">zoomed out view of all people in
the room
</ns2:description>
<ns2:priority>1</ns2:priority>
<ns2:lang>it</ns2:lang>
<ns2:mobility>static</ns2:mobility>
<ns2:view>room</ns2:view>
<ns2:capturedPeople>
  <ns2:personIDREF>alice</ns2:personIDREF>
  <ns2:personIDREF>bob</ns2:personIDREF>
  <ns2:personIDREF>ciccio</ns2:personIDREF>
</ns2:capturedPeople>
</ns2:mediaCapture>
</mediaCaptures>
<encodingGroups>
  <ns2:encodingGroup encodingGroupID="EG0">
    <ns2:maxGroupBandwidth>600000</ns2:maxGroupBandwidth>
    <ns2:encodingIDList>
      <ns2:encID>ENC1</ns2:encID>
      <ns2:encID>ENC2</ns2:encID>
      <ns2:encID>ENC3</ns2:encID>
    </ns2:encodingIDList>
  </ns2:encodingGroup>
  <ns2:encodingGroup encodingGroupID="EG1">
    <ns2:maxGroupBandwidth>300000</ns2:maxGroupBandwidth>
    <ns2:encodingIDList>
      <ns2:encID>ENC4</ns2:encID>
      <ns2:encID>ENC5</ns2:encID>
    </ns2:encodingIDList>
  </ns2:encodingGroup>
</encodingGroups>
<captureScenes>
  <ns2:captureScene scale="unknown" sceneID="CS1">
    <ns2:sceneViews>
      <ns2:sceneView sceneViewID="SE1">
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC0</ns2:captureIDREF>
          <ns2:captureIDREF>VC1</ns2:captureIDREF>
          <ns2:captureIDREF>VC2</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneView>
    </ns2:sceneViews>
  </ns2:captureScene>
</captureScenes>
```



```
        </ns2:mediaCaptureIDs>
      </ns2:sceneView>
      <ns2:sceneView sceneViewID="SE2">
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC3</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneView>
      <ns2:sceneView sceneViewID="SE3">
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC4</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneView>
      <ns2:sceneView sceneViewID="SE4">
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC4</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneView>
    </ns2:sceneViews>
  </ns2:captureScene>
</captureScenes>
<simultaneousSets>
  <ns2:simultaneousSet setID="SS1">
    <ns2:mediaCaptureIDREF>VC3</ns2:mediaCaptureIDREF>
    <ns2:sceneViewIDREF>SE1</ns2:sceneViewIDREF>
  </ns2:simultaneousSet>
  <ns2:simultaneousSet setID="SS2">
    <ns2:mediaCaptureIDREF>VC0</ns2:mediaCaptureIDREF>
    <ns2:mediaCaptureIDREF>VC2</ns2:mediaCaptureIDREF>
    <ns2:mediaCaptureIDREF>VC4</ns2:mediaCaptureIDREF>
    <ns2:mediaCaptureIDREF>VC3</ns2:mediaCaptureIDREF>
  </ns2:simultaneousSet>
</simultaneousSets>
<people>
  <ns2:person personID="bob">
    <ns2:personInfo>
      <ns3:fn>
        <ns3:text>Bob</ns3:text>
      </ns3:fn>
    </ns2:personInfo>
    <ns2:personType>minute taker</ns2:personType>
  </ns2:person>
  <ns2:person personID="alice">
    <ns2:personInfo>
      <ns3:fn>
        <ns3:text>Alice</ns3:text>
      </ns3:fn>
    </ns2:personInfo>
    <ns2:personType>presenter</ns2:personType>
  </ns2:person>
</people>
```

```

    </ns2:person>
    <ns2:person personID="ciccio">
      <ns2:personInfo>
        <ns3:fn>
          <ns3:text>Ciccio</ns3:text>
        </ns3:fn>
      </ns2:personInfo>
      <ns2:personType>chairman</ns2:personType>
      <ns2:personType>timekeeper</ns2:personType>
    </ns2:person>
  </people>
</advertisement>

```

10.2. ADV with MCCs

The associated Media Provider's telepresence capabilities are described in [I-D.ietf-clue-data-model-schema], Section "MCC example".

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<advertisement xmlns="urn:ietf:params:xml:ns:clue-protocol"
  xmlns:ns2="urn:ietf:params:xml:ns:clue-info"
  xmlns:ns3="urn:ietf:params:xml:ns:vcard-4.0" protocol="CLUE" v="0.4">
  <clueId>Napoli CLUE Endpoint</clueId>
  <sequenceNr>34</sequenceNr>
  <mediaCaptures>
    <ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="ns2:videoCaptureType" mediaType="video" captureID="AC0">
      <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
      <ns2:encGroupIDREF>EG1</ns2:encGroupIDREF>
      <ns2:spatialInformation>
        <ns2:capturePoint>
          <ns2:x>0.5</ns2:x>
          <ns2:y>1.0</ns2:y>
          <ns2:z>0.5</ns2:z>
          <ns2:lineOfCapturePoint>
            <ns2:x>0.5</ns2:x>
            <ns2:y>0.0</ns2:y>
            <ns2:z>0.5</ns2:z>
          </ns2:lineOfCapturePoint>
        </ns2:capturePoint>
      </ns2:spatialInformation>
      <ns2:individual>true</ns2:individual>
      <ns2:description lang="en">main audio from the room</ns2:description>
    </ns2:mediaCapture>
  </mediaCaptures>
</advertisement>

```

```
<ns2:priority>1</ns2:priority>
<ns2:lang>it</ns2:lang>
<ns2:mobility>static</ns2:mobility>
<ns2:view>room</ns2:view>
<ns2:capturedPeople>
  <ns2:personIDREF>alice</ns2:personIDREF>
  <ns2:personIDREF>bob</ns2:personIDREF>
  <ns2:personIDREF>ciccio</ns2:personIDREF>
</ns2:capturedPeople>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" captureID="VC0" mediaType="video" >
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
  <ns2:spatialInformation>
    <ns2:capturePoint>
      <ns2:x>0.5</ns2:x>
      <ns2:y>1.0</ns2:y>
      <ns2:z>0.5</ns2:z>
      <ns2:lineOfCapturePoint>
        <ns2:x>0.5</ns2:x>
        <ns2:y>0.0</ns2:y>
        <ns2:z>0.5</ns2:z>
      </ns2:lineOfCapturePoint>
    </ns2:capturePoint>
  </ns2:spatialInformation>
  <ns2:individual>true</ns2:individual>
  <ns2:description lang="en">left camera video capture</ns2:descriptio
n>

  <ns2:priority>1</ns2:priority>
  <ns2:lang>it</ns2:lang>
  <ns2:mobility>static</ns2:mobility>
  <ns2:view>individual</ns2:view>
  <ns2:capturedPeople>
    <ns2:personIDREF>ciccio</ns2:personIDREF>
  </ns2:capturedPeople>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" captureID="VC1" mediaType="video">
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
  <ns2:spatialInformation>
    <ns2:capturePoint>
      <ns2:x>0.5</ns2:x>
      <ns2:y>1.0</ns2:y>
      <ns2:z>0.5</ns2:z>
      <ns2:lineOfCapturePoint>
        <ns2:x>0.5</ns2:x>
        <ns2:y>0.0</ns2:y>
```

```
        <ns2:z>0.5</ns2:z>
      </ns2:lineOfCapturePoint>
    </ns2:capturePoint>
  </ns2:spatialInformation>
  <ns2:individual>true</ns2:individual>
  <ns2:description lang="en">central camera video capture
</ns2:description>
  <ns2:priority>1</ns2:priority>
  <ns2:lang>it</ns2:lang>
  <ns2:mobility>static</ns2:mobility>
  <ns2:view>individual</ns2:view>
  <ns2:capturedPeople>
    <ns2:personIDREF>alice</ns2:personIDREF>
  </ns2:capturedPeople>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC2" mediaType="video" >
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
  <ns2:spatialInformation>
    <ns2:capturePoint>
      <ns2:x>0.5</ns2:x>
      <ns2:y>1.0</ns2:y>
      <ns2:z>0.5</ns2:z>
      <ns2:lineOfCapturePoint>
        <ns2:x>0.5</ns2:x>
        <ns2:y>0.0</ns2:y>
        <ns2:z>0.5</ns2:z>
      </ns2:lineOfCapturePoint>
    </ns2:capturePoint>
  </ns2:spatialInformation>
  <ns2:individual>true</ns2:individual>
  <ns2:description lang="en">right camera video capture
</ns2:description>
  <ns2:priority>1</ns2:priority>
  <ns2:lang>it</ns2:lang>
  <ns2:mobility>static</ns2:mobility>
  <ns2:view>individual</ns2:view>
  <ns2:capturedPeople>
    <ns2:personIDREF>bob</ns2:personIDREF>
  </ns2:capturedPeople>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC3" mediaType="video" >
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
  <ns2:nonSpatiallyDefinable>true</ns2:nonSpatiallyDefinable>
  <ns2:content>
```

```
        <ns2:sceneViewIDREF>SE1</ns2:sceneViewIDREF>
    </ns2:content>
    <ns2:policy>Soundlevel:0</ns2:policy>
    <ns2:maxCaptures>1</ns2:maxCaptures>
    <ns2:allowSubsetChoice>false</ns2:allowSubsetChoice>
    <ns2:description lang="en">loudest room segment</ns2:description>
    <ns2:priority>1</ns2:priority>
    <ns2:lang>it</ns2:lang>
    <ns2:mobility>static</ns2:mobility>
    <ns2:view>individual</ns2:view>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC4" mediaType="video">
    <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
    <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
    <ns2:spatialInformation>
        <ns2:capturePoint>
            <ns2:x>0.5</ns2:x>
            <ns2:y>1.0</ns2:y>
            <ns2:z>0.5</ns2:z>
            <ns2:lineOfCapturePoint>
                <ns2:x>0.5</ns2:x>
                <ns2:y>0.0</ns2:y>
                <ns2:z>0.5</ns2:z>
            </ns2:lineOfCapturePoint>
        </ns2:capturePoint>
    </ns2:spatialInformation>
    <ns2:individual>true</ns2:individual>
    <ns2:description lang="en">zoomed out view of all people in the room
</ns2:description>
    <ns2:priority>1</ns2:priority>
    <ns2:lang>it</ns2:lang>
    <ns2:mobility>static</ns2:mobility>
    <ns2:view>room</ns2:view>
    <ns2:capturedPeople>
        <ns2:personIDREF>alice</ns2:personIDREF>
        <ns2:personIDREF>bob</ns2:personIDREF>
        <ns2:personIDREF>ciccio</ns2:personIDREF>
    </ns2:capturedPeople>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC5" mediaType="video">
    <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
    <ns2:nonSpatiallyDefinable>true</ns2:nonSpatiallyDefinable>
    <ns2:content>
        <ns2:sceneViewIDREF>SE1</ns2:sceneViewIDREF>
    </ns2:content>
    <ns2:policy>Soundlevel:1</ns2:policy>
```

```
<ns2:maxCaptures>1</ns2:maxCaptures>
<ns2:allowSubsetChoice>false</ns2:allowSubsetChoice>
<ns2:description lang="en">penultimate loudest room segment
</ns2:description>
<ns2:priority>1</ns2:priority>
<ns2:lang>it</ns2:lang>
<ns2:mobility>static</ns2:mobility>
<ns2:view>individual</ns2:view>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC6" mediaType="video">
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:nonSpatiallyDefinable>true</ns2:nonSpatiallyDefinable>
  <ns2:content>
    <ns2:sceneViewIDREF>SE1</ns2:sceneViewIDREF>
  </ns2:content>
  <ns2:composed>false</ns2:composed>
  <ns2:switched>true</ns2:switched>
  <ns2:policy>Soundlevel:2</ns2:policy>
  <ns2:maxCaptures>1</ns2:maxCaptures>
  <ns2:allowSubsetChoice>false</ns2:allowSubsetChoice>
  <ns2:description lang="en">last but two loudest room segment
  </ns2:description>
  <ns2:priority>1</ns2:priority>
  <ns2:lang>it</ns2:lang>
  <ns2:mobility>static</ns2:mobility>
  <ns2:view>individual</ns2:view>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC7" mediaType="video">
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:nonSpatiallyDefinable>true</ns2:nonSpatiallyDefinable>
  <ns2:content>
    <ns2:captureIDREF>VC3</ns2:captureIDREF>
    <ns2:captureIDREF>VC5</ns2:captureIDREF>
    <ns2:captureIDREF>VC6</ns2:captureIDREF>
  </ns2:content>
  <ns2:composed>true</ns2:composed>
  <ns2:switched>true</ns2:switched>
  <ns2:maxCaptures>1</ns2:maxCaptures>
  <ns2:allowSubsetChoice>false</ns2:allowSubsetChoice>
  <ns2:description lang="en">big picture of the current speaker +
  pips about previous speakers</ns2:description>
  <ns2:priority>1</ns2:priority>
  <ns2:lang>it</ns2:lang>
  <ns2:mobility>static</ns2:mobility>
  <ns2:view>individual</ns2:view>
</ns2:mediaCapture>
```

```
</mediaCaptures>
<encodingGroups>
  <ns2:encodingGroup encodingGroupID="EG0">
    <ns2:maxGroupBandwidth>600000</ns2:maxGroupBandwidth>
    <ns2:encodingIDList>
      <ns2:encID>ENC1</ns2:encID>
      <ns2:encID>ENC2</ns2:encID>
      <ns2:encID>ENC3</ns2:encID>
    </ns2:encodingIDList>
  </ns2:encodingGroup>
  <ns2:encodingGroup encodingGroupID="EG1">
    <ns2:maxGroupBandwidth>300000</ns2:maxGroupBandwidth>
    <ns2:encodingIDList>
      <ns2:encID>ENC4</ns2:encID>
      <ns2:encID>ENC5</ns2:encID>
    </ns2:encodingIDList>
  </ns2:encodingGroup>
</encodingGroups>
<captureScenes>
  <ns2:captureScene scale="unknown" sceneID="CS1">
    <ns2:sceneViews>
      <ns2:sceneView sceneViewID="SE1">
        <ns2:description lang="en">participants' individual videos
        </ns2:description>
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC0</ns2:captureIDREF>
          <ns2:captureIDREF>VC1</ns2:captureIDREF>
          <ns2:captureIDREF>VC2</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneView>
      <ns2:sceneView sceneViewID="SE2">
        <ns2:description lang="en">loudest segment of the room
        </ns2:description>
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC3</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneView>
      <ns2:sceneView sceneViewID="SE5">
        <ns2:description lang="en">loudest segment
        of the room + pips</ns2:description>
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC7</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneView>
      <ns2:sceneView sceneViewID="SE4">
        <ns2:description lang="en">room audio</ns2:description>
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>AC0</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneView>
    </ns2:sceneViews>
  </ns2:captureScene>
</captureScenes>
```

```
        </ns2:mediaCaptureIDs>
      </ns2:sceneView>
      <ns2:sceneView sceneViewID="SE3">
        <ns2:description lang="en">room video</ns2:description>
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC4</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneView>
    </ns2:sceneViews>
  </ns2:captureScene>
</captureScenes>
<simultaneousSets>
  <ns2:simultaneousSet setID="SS1">
    <ns2:mediaCaptureIDREF>VC7</ns2:mediaCaptureIDREF>
    <ns2:sceneViewIDREF>SE1</ns2:sceneViewIDREF>
  </ns2:simultaneousSet>
  <ns2:simultaneousSet setID="SS2">
    <ns2:mediaCaptureIDREF>VC0</ns2:mediaCaptureIDREF>
    <ns2:mediaCaptureIDREF>VC2</ns2:mediaCaptureIDREF>
    <ns2:mediaCaptureIDREF>VC4</ns2:mediaCaptureIDREF>
    <ns2:mediaCaptureIDREF>VC7</ns2:mediaCaptureIDREF>
  </ns2:simultaneousSet>
</simultaneousSets>
<people>
  <ns2:person personID="bob">
    <ns2:personInfo>
      <ns3:fn>
        <ns3:text>Bob</ns3:text>
      </ns3:fn>
    </ns2:personInfo>
    <ns2:personType>minute taker</ns2:personType>
  </ns2:person>
  <ns2:person personID="alice">
    <ns2:personInfo>
      <ns3:fn>
        <ns3:text>Alice</ns3:text>
      </ns3:fn>
    </ns2:personInfo>
    <ns2:personType>presenter</ns2:personType>
  </ns2:person>
  <ns2:person personID="ciccio">
    <ns2:personInfo>
      <ns3:fn>
        <ns3:text>Ciccio</ns3:text>
      </ns3:fn>
    </ns2:personInfo>
    <ns2:personType>chairman</ns2:personType>
    <ns2:personType>timekeeper</ns2:personType>
  </ns2:person>
</people>
```



```
        </ns2:person>
    </people>
</advertisement>
```

11. IANA Considerations

This document registers a new XML namespace, a new XML schema and the MIME type for the schema. This document also registers the "CLUE" Application Service tag and the "CLUE" Application Protocol tag and defines registries for the CLUE messages and response codes.

11.1. URN Sub-Namespace Registration

This section registers a new XML namespace,
"urn:ietf:params:xml:ns:clue-protocol".

URI: urn:ietf:params:xml:ns:clue-protocol

Registrant Contact: IETF CLUE working group (clue@ietf.org), Simon
Pietro Romano (spromano@unina.it).

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>CLUE Messages</title>
  </head>
  <body>
    <h1>Namespace for CLUE Messages</h1>
    <h2>urn:ietf:params:xml:ns:clue-protocol</h2>
    [[NOTE TO IANA/RFC-EDITOR: Please update RFC URL and replace XXXX
      with the RFC number for this specification.]]
    <p>See <a href="[[RFC URL]]">
      RFCXXXX</a>.</p>
  </body>
</html>
END
```

11.2. XML Schema registration

This section registers an XML schema per the guidelines in [RFC3688].

URI: urn:ietf:params:xml:schema:clue-protocol

Registrant Contact: CLUE working group (clue@ietf.org), Simon Pietro Romano (sromano@unina.it).

Schema: The XML for this schema can be found as the entirety of Section 9 of this document.

11.3. MIME Media Type Registration for 'application/clue+xml'

This section registers the "application/clue+xml" MIME type.

To: ietf-types@iana.org

Subject: Registration of MIME media type application/clue+xml

MIME media type name: application

MIME subtype name: clue+xml

Required parameters: (none)

Optional parameters: charset

Same as the charset parameter of "application/xml" as specified in [RFC3023], Section 3.2.

Encoding considerations: Same as the encoding considerations of "application/xml" as specified in [RFC3023], Section 3.2.

Security considerations: This content type is designed to carry protocol data related to telepresence session control. Some of the data could be considered private. This media type does not provide any protection and thus other mechanisms such as those described in Section Security are required to protect the data. This media type does not contain executable content.

Interoperability considerations: None.

Published specification: RFC XXXX [[NOTE TO IANA/RFC-EDITOR: Please replace XXXX with the RFC number for this specification.]]

Applications that use this media type: CLUE participants.

Additional Information: Magic Number(s): (none),

File extension(s): .clue,
Macintosh File Type Code(s): TEXT.

Person & email address to contact for further information: Simon
Pietro Romano (spromano@unina.it).

Intended usage: LIMITED USE

Author/Change controller: The IETF

Other information: This media type is a specialization of
application/xml [RFC3023], and many of the considerations described
there also apply to application/clue+xml.

11.4. DNS Registrations

Section 11.4.1 defines an Application Service tag of "CLUE", which is
used to identify the CLUE service. The Application Protocol tag
"CLUE", defined in Section 11.4.2, is used to identify a CLUE
Participant that understands CLUE.

11.4.1. Application Service tag

This section registers a new S-NAPTR/U-NAPTR Application Service tag
for CLUE, as mandated by [RFC3958].

Application Service Tag: CLUE

Intended usage: Identifies a server that supports CLUE telepresence
conferencing.

Defining publication: RFCXXXX [[NOTE TO IANA/RFC-EDITOR: Please
replace XXXX with the RFC number for this specification.]]

Contact information: The authors of this document

Author/Change controller: The IESG

11.4.2. Application Protocol tag

This section registers a new S-NAPTR/U-NAPTR Application Protocol tag
for CLUE, as mandated by [RFC3958].

Application Service Tag: CLUE

Intended Usage: Identifies the CLUE Protocol.

Applicable Service Tag(s): CLUE

Terminal NAPTR Record Type(s): U

Defining Publication: RFC XXXX [[NOTE TO IANA/RFC-EDITOR: Please replace XXXX with the RFC number for this specification.]]

Contact Information: The authors of this document

Author/Change Controller: The IESG

11.5. CLUE Protocol Registry

The document requests that the IANA creates new registries for CLUE messages and response codes.

11.5.1. CLUE Message Types

The following summarizes the registry for CLUE messages:

Related Registry: CLUE Message Types Registry

Defining RFC: RFC XXXX [[NOTE TO IANA/RFC-EDITOR: Please replace XXXX with the RFC number for this specification.]]

Registration/Assignment Procedures: Following the policies outlined in [RFC5226], the IANA policy for assigning new values for the CLUE message types for the CLUE protocol is Specification Required.

Registrant Contact: IETF CLUE working group (clue@ietf.org), Simon Pietro Romano (spromano@unina.it).

The initial Message table is populated using the CLUE messages described in Section 4 and defined in the XML schema in Section 9.

- o OPTIONS
- o OPTIONS RESPONSE
- o ADVERTISEMENT (ADV)
- o ADVERTISEMENT ACKNOWLEDGEMENT (ACK)
- o CONFIGURE (CONF)
- o CONFIGURE RESPONSE (CONF RESPONSE)

11.5.2. CLUE Response Codes

The following summarizes the requested registry for CLUE response codes:

Related Registry: CLUE Response Code Registry

Defining RFC: RFC XXXX [[NOTE TO IANA/RFC-EDITOR: Please replace XXXX with the RFC number for this specification.]]

Registration/Assignment Procedures: Following the policies outlined in [RFC5226], the IANA policy for assigning new values for the Response codes for CLUE shall be Specification Required.

Registrant Contact: IETF CLUE working group (clue@ietf.org), Simon Pietro Romano (spromano@unina.it).

The initial Response-code table is populated using the Response codes defined in Section 4.7 as follows:

Response code	Response string	Description
200	Success	The request has been successfully processed.
300	Bad syntax	The XML syntax of the message is not correct.
301	Invalid value	The message contains an invalid parameter value.
302	Conflicting values	The message contains values that cannot be used together.

400	Version not supported	The protocol version used in the message is not supported.
401	Invalid sequencing	The sequence number of the message is out of date.
402	Invalid identifier	The identifier used in the message is not valid or unknown.
403	ADV Expired	The number of the ADV the CONF refers to is out of date.
404	Subset choice not allowed	The subset choice is not allowed for the specified MCC

12. Diff with draft-ietf-clue-protocol-03

- o Response codes section updated.
- o maxCaptureEncodings removed from examples, allowSubsetChoice added.
- o State machines reviewed and related descriptions aligned with the diagrams.
- o Applied recommended updates indicated in Christian's review (2015-03-19).

13. Diff with draft-ietf-clue-protocol-02

- o CLUE Participant state machine: TERMINATED state replaced with IDLE.

- o MP and MC state machines: SDP O/A state removed.
- o Diff mechanism (and related example) removed.
- o Schema updates: `versionType` used as the data type for all versions fields, `xs:unsignedInt` used as the data type for all sequence number fields, diff support removed from the ADV definition.

14. Diff with draft-ietf-clue-protocol-01

- o The diff mechanism for the ADV message has been introduced.
- o READV and READV RESPONSE message have been both removed.
- o The state machines have been deeply reviewed and changed.
- o References: references have been updated and splitted into Informative references and Normative references as in framework v17.
- o Schema: `<globalSceneEntries>` changed in `<globalViews>`, `<participants>` in `<people>`
- o Terminology: many definitions added.
- o Response codes updated.

15. Diff with draft-ietf-clue-protocol-00

1. The XML schema of the ADVERTISEMENT and of the READV have been aligned with the current definitions in [I-D.ietf-clue-data-model-schema] (example of updates: `<participants>` --> `<people>`, `<globalCaptureEntries>` --> `<globalSceneEntries>`)
2. Text has been added to clarify that, in the OPTIONS RESPONSE, when the response code is not an error response code, both `<mediaProvider>` and `<mediaConsumer>` are mandatory.
3. The content of the "v" attribute and of the `<version>` elements carried in the OPTIONS and OPTIONS RESPONSE messages has been described more precisely.
4. Advertisement examples have been added.

16. Diff with draft-presta-clue-protocol-04

1. The response code type error in the OPTIONS response (and in other parts) has been corrected.

17. Diff with draft-presta-clue-protocol-03

1. The XML Schema has been deeply revised and completed.
2. The descriptions of the CLUE messages have been added.
3. The distinction between major version numbers and minor version numbers has been cut and pasted from [I-D.ietf-clue-signaling].
4. Besides the two way one, a three way mechanism for the options negotiation has been proposed and provided to foster discussion.

18. Diff with draft-presta-clue-protocol-02

1. "Terminology" section added.
2. Introduced the concept of "CLUE Participant" - an Endpoint or a MCU able to use the CLUE protocol within a telepresence session. A CLUE Participant can act as a Media Provider and/or as a Media Consumer.
3. Introduced the ACK/NACK mechanism for the ADVERTISEMENT.
4. MP and MC state machines have been updated. The CP state machine has been added.

19. Acknowledgments

The authors thank all the CLUErs for their precious feedbacks and support, in particular Paul Kyzivat, Christian Groves and Scarlett Liuyan.

20. References

20.1. Normative References

- | | |
|-----------------------------------|---|
| [I-D.ietf-clue-data-model-schema] | Presta, R. and S. Romano, "An XML Schema for the CLUE data model", draft-ietf-clue-data-model-schema-09 (work in progress), April 2015. |
| [I-D.ietf-clue-datachannel] | Holmberg, C., "CLUE Protocol data channel", |

draft-ietf-clue-datachannel-09
(work in progress), March 2015.

[I-D.ietf-clue-framework] Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-22 (work in progress), April 2015.

[I-D.ietf-clue-signaling] Kyzivat, P., Xiao, L., Groves, C., and R. Hansen, "CLUE Signaling", draft-ietf-clue-signaling-05 (work in progress), March 2015.

[RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC3958] Daigle, L. and A. Newton, "Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)", RFC 3958, January 2005.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

20.2. Informative References

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

[RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117,

January 2008.

- [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.
- [RFC6502] Camarillo, G., Srinivasan, S., Even, R., and J. Urpalainen, "Conference Event Package Data Format Extension for Centralized Conferencing (XCON)", RFC 6502, March 2012.
- [RFC6503] Barnes, M., Boulton, C., Romano, S., and H. Schulzrinne, "Centralized Conferencing Manipulation Protocol", RFC 6503, March 2012.
- [RFC7262] Romanow, A., Botzko, S., and M. Barnes, "Requirements for Telepresence Multistreams", RFC 7262, June 2014.

Authors' Addresses

Roberta Presta
University of Napoli
Via Claudio 21
Napoli 80125
Italy

EMail: roberta.presta@unina.it

Simon Pietro Romano
University of Napoli
Via Claudio 21
Napoli 80125
Italy

EMail: spromano@unina.it

CLUE WG
Internet-Draft
Intended status: Standards Track
Expires: April 20, 2016

R. Even
Huawei Technologies
J. Lennox
Vidyo
October 18, 2015

Mapping RTP streams to CLUE media captures
draft-ietf-clue-rtp-mapping-05.txt

Abstract

This document describes how the Real Time transport Protocol (RTP) is used in the context of the CLUE protocol. It also describes the mechanisms and recommended practice for mapping RTP media streams defined in SDP to CLUE media captures.

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 April 20, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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	2
2. Terminology	3
3. RTP topologies for CLUE	3
4. Mapping CLUE Capture Encodings to RTP streams	5
4.1. Review of RTP related documents relevant to CLUE work.	6
4.2. Requirements of a solution	7
4.3. Static Mapping	9
4.4. Dynamic mapping	9
4.5. Recommendations	9
5. Application to CLUE Media Requirements	10
6. CaptureID definition	11
6.1. RTCP CaptureId SDES Item	12
6.2. RTP Header Extension	12
7. Examples	12
8. Acknowledgements	13
9. IANA Considerations	13
10. Security Considerations	13
11. References	15
11.1. Normative References	15
11.2. Informative References	15
Authors' Addresses	17

1. Introduction

Telepresence systems can send and receive multiple media streams. The CLUE framework [I-D.ietf-clue-framework] defines media captures as a source of Media, such as from one or more Capture Devices. A Media Capture may also be constructed from other Media streams. A middle box can express conceptual Media Captures that it constructs from Media streams it receives. A Multiple Content Capture (MCC) is a special Media Capture composed of multiple Media Captures.

SIP offer answer [RFC3264] uses SDP [RFC4566] to describe the RTP[RFC3550] media streams. Each RTP stream has a unique SSRC within its RTP session. The content of the RTP stream is created by an encoder in the endpoint. This may be an original content from a camera or a content created by an intermediary device like an MCU.

This document makes recommendations, for the telepresence architecture, about how RTP and RTCP streams should be encoded and transmitted, and how their relation to CLUE Media Captures should be communicated. The proposed solution supports multiple RTP topologies.

With regards to the media (audio and video), systems that support CLUE use RTP for the media, SDP for codec and media transport

negotiation (CLUE individual encodings) and the CLUE protocol for media Capture description and selection. In order to associate the media in the different protocols there are three mapping that need to be specified:

1. CLUE individual encodings to SDP
2. RTP media streams to SDP (this is not a CLUE specific mapping)
3. RTP media streams to MC to map the received RTP steam to the current MC in the MCC.

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[RFC2119] and indicate requirement levels for compliant RTP implementations.

3. RTP topologies for CLUE

The typical RTP topologies used by Telepresence systems specify different behaviors for RTP and RTCP distribution. A number of RTP topologies are described in [I-D.ietf-avtcore-rtp-topologies-update]. For telepresence, the relevant topologies include point-to-point, as well as media mixers, media- switching mixers, and Selective Forwarding middleboxes.

In the point-to-point topology, one peer communicates directly with a single peer over unicast. There can be one or more RTP sessions, and each RTP session can carry multiple RTP streams identified by their SSRC. All SSRCs will be recognized by the peers based on the information in the RTCP SDES report that will include the CNAME and SSRC of the sent RTP streams. There are different point to point use cases as specified in CLUE use case [RFC7205]. There may be a difference between the symmetric and asymmetric use cases. While in the symmetric use case the typical mapping will be from a Media capture device to a render device (e.g. camera to monitor) in the asymmetric case the render device may receive different capture information (RTP stream from different cameras) if it has fewer rendering devices (monitors). In some cases, a CLUE session which, at a high-level, is point-to-point may nonetheless have RTP which is best described by one of the mixer topologies. For example, a CLUE endpoint can produce composite or switched captures for use by a receiving system with fewer displays than the sender has cameras. The Media capture may be described using MCC.

For the Media Mixer topology

[I-D.ietf-avtcore-rtp-topologies-update], the peers communicate only with the mixer. The mixer provides mixed or composited media streams, using its own SSRC for the sent streams. There are two cases here. In the first case the mixer may have separate RTP sessions with each peer (similar to the point to point topology) terminating the RTCP sessions on the mixer; this is known as Topo-RTCP-Terminating MCU in [I-D.ietf-avtcore-rtp-topologies-update]. In the second case, the mixer can use a conference-wide RTP session similar to [I-D.ietf-avtcore-rtp-topologies-update] Topo-mixer or Topo-Video-switching. The major difference is that for the second case, the mixer uses conference-wide RTP sessions, and distributes the RTCP reports to all the RTP session participants, enabling them to learn all the CNAMEs and SSRCs of the participants and know the contributing source or sources (CSRCs) of the original streams from the RTP header. In the first case, the Mixer terminates the RTCP and the participants cannot know all the available sources based on the RTCP information. The conference roster information including conference participants, endpoints, media and media-id (SSRC) can be available using the conference event package [RFC4575] element.

In the Media-Switching Mixer topology

[I-D.ietf-avtcore-rtp-topologies-update], the peer to mixer communication is unicast with mixer RTCP feedback. It is conceptually similar to a compositing mixer as described in the previous paragraph, except that rather than compositing or mixing multiple sources, the mixer provides one or more conceptual sources selecting one source at a time from the original sources. The Mixer creates a conference-wide RTP session by sharing remote SSRC values as CSRCs to all conference participants.

In the Selective Forwarding middlebox topology, the peer to mixer communication is unicast with RTCP mixer feedback. Every potential sender in the conference has a source which may be "projected" by the mixer into every other RTP session in the conference; thus, every original source is maintained with an independent RTP identity to every receiver, maintaining separate decoding state and its original RTCP SDES information. However, RTCP is terminated at the mixer, which might also perform reliability, repair, rate adaptation, or transcoding on the stream. Senders' SSRCs may be renumbered by the mixer. The sender may turn the projected sources on and off at any time, depending on which sources it thinks are most relevant for the receiver; this is the primary reason why this topology must act as an RTP mixer rather than as a translator, as otherwise these disabled sources would appear to have enormous packet loss. Source switching is accomplished through this process of enabling and disabling projected sources, with the higher-level semantic assignment of reason for the RTP streams assigned externally.

The above topologies demonstrate two major RTP/RTCP behaviors:

1. The mixer may either use the source SSRC when forwarding RTP packets, or use its own created SSRC. Still the mixer will distribute all RTCP information to all participants creating conference-wide RTP session/s. This allows the participants to learn the available RTP sources in each RTP session. The original source information will be the SSRC or in the CSRC depending on the topology. The point to point case behaves like this.
 2. The mixer terminates the RTCP from the source, creating separate RTP sessions with the peers. In this case the participants will not receive the source SSRC in the CSRC. Since this is usually a mixer topology, the source information is available from the SIP conference event package [RFC4575]. Subscribing to the conference event package allows each participant to know the SSRCs of all sources in the conference.
4. Mapping CLUE Capture Encodings to RTP streams

The different topologies described in Section 3 create different SSRC distribution models and RTP stream multiplexing points.

Most video conferencing systems today can separate multiple RTP sources by placing them into separate RTP sessions using, the SDP description. For example, main and slides video sources are separated into separate RTP sessions based on the content attribute [RFC4796]. This solution works straightforward if the multiplexing point is at the UDP transport level, where each RTP stream uses a separate RTP session. This will also be true for mapping the RTP streams to Media Captures Encodings if each media capture encodings uses a separate RTP session, and the consumer can identify it based on the receiving RTP port. In this case, SDP only needs to label the RTP session with an identifier that can be used to identify the media capture in the CLUE description. The SDP label attribute serves as this identifier. In this case, the mapping does not change even if the RTP session is switched using same or different SSRC. (The multiplexing is not at the SSRC level).

Even though Session multiplexing is supported by CLUE, for scaling reasons, CLUE recommends using SSRC multiplexing in a single or multiple sessions using [I-D.ietf-mmusic-sdp-bundle-negotiation]. So we need to look at how to map RTP streams to Captures Encodings when SSRC multiplexing is used.

When looking at SSRC multiplexing we can see that in various topologies, the SSRC behavior may be different:

1. The SSRCs are static (assigned by the MCU/Mixer), and there is an SSRC for each media capture encoding defined in the CLUE protocol. Source information may be conveyed using CSRC, or, in the case of topo-RTCP-Terminating MCU, is not conveyed.
2. The SSRCs are dynamic, representing the original source and are relayed by the Mixer/MCU to the participants.

In the above two cases the MCU/Mixer may create an advertisement, with a virtual room capture scene.

Another case we can envision is that the MCU / Mixer relays all the capture scenes from all advertisements to all consumers. This means that the advertisement will include multiple capture scenes, each representing a separate TelePresence room with its own coordinate system.

MCCs bring another mapping issue, in that an MCC represents multiple Media Captures that can be sent as part of this MCC if configured by the consumer. When receiving an RTP stream which is mapped to the MCC, the consumer needs to know which original MC it is in order to get the MC parameters from the advertisement. If a consumer requested a MCC, the original MC does not have a capture encoding, so it cannot be associated with an m-line using a label as described in CLUE signaling [I-D.ietf-clue-signaling]. This is important, for example, to get correct scaling information for the original MC, which may be different for the various MCs that are contributing to the MCC.

4.1. Review of RTP related documents relevant to CLUE work.

Editor's note: This section provides an overview of the RFCs and drafts that are can be used in a CLUE system and as a base for a mapping solution. This section is for information only; the normative behavior is given in the cited documents. Tools for SSRC multiplexing support are defined for general conferencing applications; CLUE systems use the same tools.

When looking at the available tools based on current work in MMUSIC, AVTcore and AVText for supporting SSRC multiplexing the following documents are considered to be relevant.

Negotiating Media Multiplexing Using the Session Description Protocol in [I-D.ietf-mmusic-sdp-bundle-negotiation] defines a "bundle" SDP grouping extension that can be used with SDP Offer/Answer mechanism to negotiate the usage of a single 5-tuple for sending and receiving media associated with multiple SDP media descriptions ("m="). [bundle] specifies how to associate a received RTP stream with the

m-line describing it. The assumption in the work is that each SDP m-line represents a single media source.

[I-D.ietf-mmusic-sdp-bundle-negotiation] specifies using the SDP mid value and sending it as RTCP SDES and an RTP header extension in order to be able to map the RTP stream to the SDP m-line. This is relevant when there are multiple RTP streams with the same payload subtype number.

SDP Source attribute [RFC5576] mechanisms to describe specific attributes of RTP sources based on their SSRC.

Negotiation of generic image attributes in SDP [RFC6236] provides the means to negotiate the image size. The image attribute can be used to offer different image parameters like size but in order to offer multiple RTP streams with different resolutions it does it using separate RTP session for each image option

([I-D.ietf-mmusic-sdp-bundle-negotiation] provides the support of a single RTP session but each image option will need a separate SDP m-line).

The recommended support of the simulcast case is to use [I-D.ietf-mmusic-sdp-simulcast]

In the next sections, the document will propose mechanisms to map the RTP streams to media captures addressing.

4.2. Requirements of a solution

This section lists, more briefly, the requirements a media architecture for Clue telepresence needs to achieve, summarizing the discussion of previous sections. In this section, RFC 2119 [RFC2119] language refers to requirements on a solution, not an implementation; thus, requirements keywords are not written in capital letters.

Media-1: It must not be necessary for a Clue session to use more than a single transport flow for transport of a given media type (video or audio).

Media-2: It must, however, be possible for a Clue session to use multiple transport flows for a given media type where it is considered valuable (for example, for distributed media, or differential quality-of-service).

Media-3: It must be possible for a Clue endpoint or MCU to simultaneously send sources corresponding to static captures and to both composited and switched multi-content captures in the same transport flow. (Any given device might not necessarily be able send

all of these source types; but for those that can, it must be possible for them to be sent simultaneously.)

Media-4: It must be possible for an original source to move among multi-content captures (i.e. at one time be sent for one MCC, and at a later time be sent for another one).

Media-5: It must be possible for a source to be placed into a MCC even if the source is a "late joiner", i.e. was added to the conference after the receiver requested the MCC.

Media-6: Whenever a given source is assigned to a switched capture, it must be immediately possible for a receiver to determine the MCC it corresponds to, and thus that any previous source is no longer being mapped to that switched capture.

Media-7: It must be possible for a receiver to identify the original capture(s) that are currently being mapped to an MCC, and correlate it with both the Clue advertisement and out-of-band (non-Clue) information such as rosters.

Media-8: It must be possible for a source to move among MCCs without requiring a refresh of decoder state (e.g., for video, a fresh I-frame), when this is unnecessary. However, it must also be possible for a receiver to indicate when a refresh of decoder state is in fact necessary.

Media-9: If a given source is being sent on the same transport flow for more than one reason (e.g. if it corresponds to more than one switched capture at once, or to a static capture), it should be possible for a sender to send only one copy of the source.

Media-10: On the network, media flows should, as much as possible, look and behave like currently-defined usages of existing protocols; established semantics of existing protocols must not be redefined.

Media-11: The solution should seek to minimize the processing burden for boxes that distribute media to decoding hardware.

Media-12: If multiple sources from a single synchronization context are being sent simultaneously, it must be possible for a receiver to associate and synchronize them properly, even for sources that are mapped to switched captures.

4.3. Static Mapping

Static mapping is widely used in current MCU implementations. It is also common for a point to point symmetric use case when both endpoints have the same capabilities. For capture encodings with static SSRCS, it is most straightforward to indicate this mapping outside the media stream, in the CLUE or SDP signaling. When using SSRC multiplexing [I-D.ietf-mmusic-sdp-bundle-negotiation] defines the use of the SDP mid attribute value to associate between the received RTP stream and the SDP m-line. The mid is carried as an RTP header extension and RTCP SDES message defined in [I-D.ietf-mmusic-sdp-bundle-negotiation] .

4.4. Dynamic mapping

Dynamic mapping by tagging each media packet with the SDP mid value. This means that a receiver immediately knows how to interpret received media, even when an unknown SSRC is seen. As long as the media carries a known mid, it can be assumed that this media stream will replace the stream currently being received with that mid.

This gives significant advantages to switching latency, as a switch between sources can be achieved without any form of negotiation with the receiver.

However, the disadvantage in using a mid in the stream that it introduces additional processing costs for every media packet, as mid are scoped only within one hop (i.e., within a cascaded conference a mid that is used from the source to the first MCU is not meaningful between two MCUs, or between an MCU and a receiver), and so they may need to be added or modified at every stage.

An additional issue with putting mid in the RTP packets comes from cases where a non-bundle aware endpoint is being switched by an MCU to a bundle endpoint. In this case, we may require up to an additional 12 bytes in the RTP header, which may push a media packet over the MTU. However, as the MTU on either side of the switch may not match, it is possible that this could happen even without adding extra data into the RTP packet. The 12 additional bytes per packet could also be a significant bandwidth increase in the case of very low bandwidth audio codecs.

4.5. Recommendations

The recommendation is that CLUE endpoint using SSRC multiplexing MUST support [[I-D.ietf-mmusic-sdp-bundle-negotiation] and use the SDP mid attribute for mapping.

5. Application to CLUE Media Requirements

The requirement section Section 4.2 offers a number of requirements that are believed to be necessary for a CLUE RTP mapping. The solutions described in this document are believed to meet these requirements, though some of them are only possible for some of the topologies. (Since the requirements are generally of the form "it must be possible for a sender to do something", this is adequate; a sender which wishes to perform that action needs to choose a topology which allows the behavior it wants.

In this section we address only those requirements where the topologies or the association mechanisms treat the requirements differently.

Media-4: It must be possible for an original source to move among switched captures (i.e. at one time be sent for one switched capture, and at a later time be sent for another one).

This applies naturally for static sources with a Switched Mixer. For dynamic sources with a Selective Forwarding middlebox, this just requires the mid in the header extension element to be updated appropriately.

Media-6: Whenever a given source is transmitted for a switched capture, it must be immediately possible for a receiver to determine the switched capture it corresponds to, and thus that any previous source is no longer being mapped to that switched capture.

For a Switched Mixer, this applies naturally. For a Selective Forwarding middlebox, this is done based on the mid.

Media-7: It must be possible for a receiver to identify the original source that is currently being mapped to a switched capture, and correlate it with out-of-band (non-Clue) information such as rosters.

For a Switched Mixer, this is done based on the CSRC, if the mixer is providing CSRCs; For a Selective Forwarding middlebox, this is done based on the SSRC.

For MCC which can represent multiple switched MCs there is a need to know which MC represents the current RTP stream, requires a mapping from an RTP stream to an MC. In order to address this mapping this document defines an RTP header extension that includes the CaptureID in order to map to the original MC allowing the consumer to use the MC attributes like the spatial information.

Media-8: It must be possible for a source to move among switched captures without requiring a refresh of decoder state (e.g., for video, a fresh I-frame), when this is unnecessary. However, it must also be possible for a receiver to indicate when a refresh of decoder state is in fact necessary.

This can be done by a Selective Forwarding middlebox, but not by a Switching Mixer. The last requirement can be accomplished through an FIR message [RFC5104], though potentially a faster mechanism (not requiring a round-trip time from the receiver) would be preferable.

Media-9: If a given source is being sent on the same transport flow to satisfy more than one capture (e.g. if it corresponds to more than one switched capture at once, or to a static capture as well as a switched capture), it should be possible for a sender to send only one copy of the source.

For a Selective Forwarding middlebox, this may be a problem since an encoding can be used by a single MC, it will require using the same SDP label for multiple MC (example middle camera and active speaker MC) this can also be done for an environment with a hybrid of mixer topologies and static and dynamic captures. It is not possible for static captures from a Switched Mixer.

Media-12: If multiple sources from a single synchronization context are being sent simultaneously, it must be possible for a receiver to associate and synchronize them properly, even for sources that are mapped to switched captures.

For a Mixed or Switched Mixer topology, receivers will see only a single synchronization context (CNAME), corresponding to the mixer. For a Selective Forwarding middlebox, separate projecting sources keep separate synchronization contexts based on their original CNAMEs, thus allowing independent synchronization of sources from independent rooms without needing global synchronization. In hybrid cases, however (e.g. if audio is mixed), all sources which need to be synchronized with the mixed audio must get the same CNAME (and thus a mixer-provided timebase) as the mixed audio.

6. CaptureID definition

For mapping an RTP stream to a specific MC in the MCC the CLUE captureId is used. The media sender MUST send for MCC the captureID in the RTP header and as a RTCP SDP message.

6.1. RTCP CaptureId SDES Item

This document specifies a new RTCP SDES message

```

0          1          2          3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-----+-----+-----+-----+-----+-----+-----+-----+
  | CaptureId = XXX |      length      | CaptureId
  +-----+-----+-----+-----+-----+-----+-----+-----+
  |      ....

```

This CaptureID is the same as in the CLUE MC and is also used in the RTP header extension.

This SDES message MAY be sent in a compound RTCP packet based on the application need.

6.2. RTP Header Extension

The CaptureId is carried within the RTP header extension field, using [RFC5285] two bytes header extension.

Support is negotiated within the SDP, i.e.

```
a=extmap:1 urn:ietf:params:rtp-hdrex:CaptureId
```

Packets tagged by the sender with the CapturId then contain a header extension as shown below

```

0          1          2          3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-----+-----+-----+-----+-----+-----+-----+-----+
  |  ID      |      Len-1      |      CaptureId
  +-----+-----+-----+-----+-----+-----+-----+-----+
  | CaptureId ..      |
  +-----+-----+-----+

```

There is no need to send the CaptureId header extension with all RTP packets. Senders MAY choose to send it only when a new MC is sent. If such a mode is being used, the header extension SHOULD be sent in the first few RTP packets to reduce the risk of losing it due to packet loss.

7. Examples

TBD

8. Acknowledgements

The authors would like to thanks Allyn Romanow and Paul Witty for contributing text to this work.

9. IANA Considerations

This document defines a new extension URI in the RTP Compact Header Extensions subregistry of the Real-Time Transport Protocol (RTP) Parameters registry, according to the following data:

Extension URI: urn:ietf:params:rtp-hdext:CaptureId

Description: CLUE CaptureId

Contact: roni.even@mail01.huawei.com

Reference: RFC XXXX

The IANA is requested to register one new RTCP SDES items in the "RTCP SDES Item Types" registry, as follows:

Value	Abbrev	Name	Reference
TBA	CCID	CLUE CaptureId	[RFCXXXX]

10. Security Considerations

The security considerations of the RTP specification, the RTP/SAVPF profile, and the various RTP/RTCP extensions and RTP payload formats that form the complete protocol suite described in this memo apply. It is not believed there are any new security considerations resulting from the combination of these various protocol extensions.

The Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback [RFC5124] (RTP/SAVPF) provides handling of fundamental issues by offering confidentiality, integrity and partial source authentication. A mandatory to support media security solution is created by combining this secured RTP profile and DTLS-SRTP keying [RFC5764]

RTCP packets convey a Canonical Name (CNAME) identifier that is used to associate RTP packet streams that need to be synchronised across related RTP sessions. Inappropriate choice of CNAME values can be a privacy concern, since long-term persistent CNAME identifiers can be used to track users across multiple calls. This memo mandates generation of short-term persistent RTCP CNAMEs, as specified in RFC7022 [RFC7022], resulting in untraceable CNAME values that alleviate this risk.

Some potential denial of service attacks exist if the RTCP reporting interval is configured to an inappropriate value. This could be done by configuring the RTCP bandwidth fraction to an excessively large or small value using the SDP "b=RR:" or "b=RS:" lines [RFC3556], or some similar mechanism, or by choosing an excessively large or small value for the RTP/AVPF minimal receiver report interval (if using SDP, this is the "a=rtcp-fb:... trr-int" parameter) [RFC4585]. The risks are as follows:

1. the RTCP bandwidth could be configured to make the regular reporting interval so large that effective congestion control cannot be maintained, potentially leading to denial of service due to congestion caused by the media traffic;
2. the RTCP interval could be configured to a very small value, causing endpoints to generate high rate RTCP traffic, potentially leading to denial of service due to the non-congestion controlled RTCP traffic; and
3. RTCP parameters could be configured differently for each endpoint, with some of the endpoints using a large reporting interval and some using a smaller interval, leading to denial of service due to premature participant timeouts due to mismatched timeout periods which are based on the reporting interval (this is a particular concern if endpoints use a small but non-zero value for the RTP/AVPF minimal receiver report interval (trr-int) [RFC4585], as discussed in [I-D.ietf-avtcore-rtp-multi-stream]).

Premature participant timeout can be avoided by using the fixed (non-reduced) minimum interval when calculating the participant timeout ([I-D.ietf-avtcore-rtp-multi-stream]). To address the other concerns, endpoints SHOULD ignore parameters that configure the RTCP reporting interval to be significantly longer than the default five second interval specified in [RFC3550] (unless the media data rate is so low that the longer reporting interval roughly corresponds to 5% of the media data rate), or that configure the RTCP reporting interval small enough that the RTCP bandwidth would exceed the media bandwidth.

The guidelines in [RFC6562] apply when using variable bit rate (VBR) audio codecs such as Opus. The use of the encryption of the header extensions are RECOMMENDED, unless there are known reasons, like RTP middleboxes performing voice activity based source selection or third party monitoring that will greatly benefit from the information, and this has been expressed using API or signalling. If further evidence are produced to show that information leakage is significant from audio level indications, then use of encryption needs to be mandated at that time.

In multi-party communication scenarios using RTP Middleboxes, a lot of trust is placed on these middleboxes to preserve the sessions security. The middlebox needs to maintain the confidentiality, integrity and perform source authentication. The middlebox can perform checks that prevents any endpoint participating in a conference to impersonate another. Some additional security considerations regarding multi-party topologies can be found in [I-D.ietf-avtcore-rtp-topologies-update]

11. References

11.1. Normative References

[I-D.ietf-clue-framework]

Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-23 (work in progress), September 2015.

[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-23 (work in progress), July 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

11.2. Informative References

[I-D.ietf-avtcore-rtp-multi-stream]

Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "Sending Multiple Media Streams in a Single RTP Session", draft-ietf-avtcore-rtp-multi-stream-09 (work in progress), September 2015.

[I-D.ietf-avtcore-rtp-topologies-update]

Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update-10 (work in progress), July 2015.

[I-D.ietf-clue-signaling]

Kyzivat, P., Xiao, L., Groves, C., and R. Hansen, "CLUE Signaling", draft-ietf-clue-signaling-06 (work in progress), August 2015.

- [I-D.ietf-mmusic-sdp-simulcast]
Westerlund, M., Nandakumar, S., and M. Zanaty, "Using Simulcast in SDP and RTP Sessions", draft-ietf-mmusic-sdp-simulcast-02 (work in progress), October 2015.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <<http://www.rfc-editor.org/info/rfc3264>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<http://www.rfc-editor.org/info/rfc3550>>.
- [RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, DOI 10.17487/RFC3556, July 2003, <<http://www.rfc-editor.org/info/rfc3556>>.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <<http://www.rfc-editor.org/info/rfc4566>>.
- [RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, Ed., "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, DOI 10.17487/RFC4575, August 2006, <<http://www.rfc-editor.org/info/rfc4575>>.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <<http://www.rfc-editor.org/info/rfc4585>>.
- [RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, DOI 10.17487/RFC4796, February 2007, <<http://www.rfc-editor.org/info/rfc4796>>.
- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <<http://www.rfc-editor.org/info/rfc5104>>.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, DOI 10.17487/RFC5117, January 2008, <<http://www.rfc-editor.org/info/rfc5117>>.

- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/RFC5124, February 2008, <<http://www.rfc-editor.org/info/rfc5124>>.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, DOI 10.17487/RFC5285, July 2008, <<http://www.rfc-editor.org/info/rfc5285>>.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, DOI 10.17487/RFC5576, June 2009, <<http://www.rfc-editor.org/info/rfc5576>>.
- [RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <<http://www.rfc-editor.org/info/rfc5764>>.
- [RFC6236] Johansson, I. and K. Jung, "Negotiation of Generic Image Attributes in the Session Description Protocol (SDP)", RFC 6236, DOI 10.17487/RFC6236, May 2011, <<http://www.rfc-editor.org/info/rfc6236>>.
- [RFC6562] Perkins, C. and JM. Valin, "Guidelines for the Use of Variable Bit Rate Audio with Secure RTP", RFC 6562, DOI 10.17487/RFC6562, March 2012, <<http://www.rfc-editor.org/info/rfc6562>>.
- [RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 7022, DOI 10.17487/RFC7022, September 2013, <<http://www.rfc-editor.org/info/rfc7022>>.
- [RFC7205] Romanow, A., Botzko, S., Duckworth, M., and R. Even, Ed., "Use Cases for Telepresence Multistreams", RFC 7205, DOI 10.17487/RFC7205, April 2014, <<http://www.rfc-editor.org/info/rfc7205>>.

Authors' Addresses

Roni Even
Huawei Technologies
Tel Aviv
Israel

Email: roni.even@mail01.huawei.com

Jonathan Lennox
Vidyo, Inc.
433 Hackensack Avenue
Seventh Floor
Hackensack, NJ 07601
US

Email: jonathan@vidyo.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 6, 2016

P. Kyzivat
L. Xiao
C. Groves
Huawei
R. Hansen
Cisco Systems
August 5, 2015

CLUE Signaling
draft-ietf-clue-signaling-06

Abstract

This document specifies how CLUE-specific signaling such as the CLUE protocol [I-D.ietf-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 February 6, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

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	3
2. Terminology	3
3. Media Feature Tag Definition	4
4. SDP Grouping Framework CLUE Extension Semantics	4
4.1. General	4
4.2. The CLUE data channel and the CLUE grouping semantic . .	4
4.3. CLUE-controlled media and the CLUE grouping semantic . .	5
4.4. SDP semantics for CLUE-controlled media	5
4.4.1. Signalling CLUE Encodings	5
4.4.1.1. Referencing Encodings in the CLUE protocol . . .	6
4.4.1.2. Media line directionality	7
4.4.2. Negotiating receipt of CLUE Capture Encodings in SDP	7
4.5. SDP Offer/Answer Procedures	7
4.5.1. Generating the Initial Offer	7
4.5.2. Generating the Answer	8
4.5.2.1. Negotiating use of CLUE and the CLUE data channel	8
4.5.2.2. Negotiating CLUE-controlled media	8
4.5.2.3. Negotiating non-CLUE controlled media	9
4.5.3. Processing the initial Offer/Answer negotiation . . .	9
4.5.3.1. Successful CLUE negotiation	9
4.5.3.2. CLUE negotiation failure	10
4.5.4. Modifying the session	10
4.5.4.1. Adding and removing CLUE-controlled media	10
4.5.4.2. Enabling CLUE mid-call	10
4.5.4.3. Disabling CLUE mid-call	11
5. Interaction of CLUE protocol and SDP negotiations	11
5.1. Independence of SDP and CLUE negotiation	11
5.2. Constraints on sending media	12
5.3. Recommendations for operating with non-atomic operations	12
6. Interaction of CLUE protocol and RTP/RTCP CaptureID	13
6.1. CaptureID reception during MCC redefinition	14
7. Multiplexing of CLUE-controlled media using BUNDLE	14
7.1. Overview	14
7.2. Usage of BUNDLE with CLUE	15
7.2.1. Generating the Initial Offer	15
7.2.2. Bundle Address Synchronization	15
7.2.3. Multiplexing of the data channel and RTP media . . .	15
8. Example: A call between two CLUE-capable Endpoints	16
9. Example: A call between a CLUE-capable and non-CLUE Endpoint	24
10. Acknowledgements	25
11. IANA Considerations	25
11.1. New SDP Grouping Framework Attribute	25

11.2. New SIP Media Feature Tag	26
12. Security Considerations	26
13. Change History	27
14. References	32
14.1. Normative References	32
14.2. Informative References	33
Authors' Addresses	34

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.ietf-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 with 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 uses terminology defined in the CLUE Framework [I-D.ietf-clue-framework].

A few additional terms specific to this document are defined as follows:

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; support for multiple CLUE groups in a single session is beyond the scope of this document. 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 signalled, 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 negotiated over SDP as described in the relevant document. A CLUE-capable device wishing to negotiate CLUE MUST also

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.

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.ietf-clue-protocol]. For an "m" line to be CLUE-controlled, its "mid" value MUST be included in a 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.

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

The restrictions on CLUE-controlled media always apply to "m" lines in an SDP offer or answer, even if negotiation of the data channel in SDP failed due to lack of CLUE support by the remote device or for any other reason, or in an offer if the recipient does not include the "mid" of the corresponding "m" line in their CLUE group.

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, unless an "m" line represents a dependent stream related to another "m" line (such as a FEC stream), in which case it MUST have the same label value as the "m" line on which it is dependent.

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> (in encodingIDListType) in a CLUE Advertisement message SHOULD represent 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 Advertisement message with a label value corresponding to the text content of the <encID>.

Similarly, each <encodingID> (in captureEncodingType) in a CLUE Configure message SHOULD represent 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 <encodingID>.

Note that the non-atomic nature of SDP/CLUE protocol interaction may mean that there are temporary periods where an <encID>/<encodingID> 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 tendency 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 the 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 MAY still include a corresponding data channel "m" line if there are any other non-CLUE protocols it can convey over that channel, but MUST NOT negotiate use of the CLUE protocol on this 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. Negotiating 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.

The receiver of the initial offer, if making the call CLUE-enabled with their SDP answer, can make their preference clear by their action in accepting or rejecting non-CLUE-controlled media lines. Rejecting these "m" lines will ensure that no non-CLUE-controlled media flows before the CLUE-controlled media is negotiated. In contrast, accepting one or more non-CLUE-controlled "m" lines in this initial answer will enable initial media to flow.

If the answerer chooses to send initial non-CLUE-controlled media in a CLUE-enabled call, Section 4.5.4.1 addresses the need to disable it once CLUE-controlled media is fully negotiated.

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.ietf-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" line 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 deactivate "m" lines for CLUE-controlled media.

Once CLUE media has been successfully negotiated devices SHOULD ensure that non-CLUE-controlled media is deactivated in cases where it corresponds to the media type of CLUE-controlled media that has been successfully negotiated. This deactivate may require an additional SDP exchange, or may be incorporated into one that is part of the CLUE negotiation.

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, an SDP offer/answer exchange completes with both sides having included a data channel "m" line in their SDP and with 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. If the data channel is still present but not included in the CLUE group semantic CLUE protocol messages MUST no longer be sent.

Note that this is distinct to cases where the CLUE data channel fails or an error occurs on the CLUE protocol; see [I-D.ietf-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 (unless the SDP negotiation has removed the CLUE channel).

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.

CLUE messages contain an <encID> (in encodingIDListType) or <encodingID> (in captureEncodingType), which is used to identify a specific encoding or captureEncoding in SDP; see [I-D.ietf-clue-data-model-schema] for specifics. 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 a CLUE protocol messages solely because it contains <encID> elements that do not match an id in SDP.

The current state of the CLUE participant or 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 ESTABLISHED to ADV (Media Provider state machine) or from ESTABLISHED to WAIT FOR CONF RESPONSE (Media Consumer state machine) based on the SDP state. See [I-D.ietf-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. Interaction of CLUE protocol and RTP/RTCP CaptureID

[I-D.ietf-clue-framework] allows for Multiple Content Captures MCCs): Captures which contain multiple source Captures, whether composited into a single stream or switched based on some metric.

The Captures that constitute these MCCs may or may not be defined in the Advertisement message. If they are defined and the MCC is providing them in a switched format the recipient may wish to determine which originating source Capture is currently being provided, so that they can apply geometric corrections based on that Capture's geometry, or take some other action based on the original Capture information.

To do this, [I-D.ietf-clue-rtp-mapping] allows for the CaptureID of the originating Capture to be conveyed via RTP or RTCP. A Media Provider sending switched media from an MCC with defined originating sources MUST send the CaptureID in both RTP and RTCP, as described in the mapping document.

6.1. CaptureID reception during MCC redefinition

Because the RTP/RTCP CaptureID is delivered via a different channel to the Advertisement in which the contents of the MCC are defined there is an intrinsic race condition in cases in which the contents of an MCC are redefined.

When a Media Provider redefines an MCC which involves CaptureIDs, the reception of the relevant CaptureIDs by the recipient will either lead or lag reception and processing of the new Advertisement by the recipient. As such, a media recipient MUST not be disrupted by any of the following in any CLUE- controlled media stream it is receiving, whether that stream is for a static Capture or for an MCC (as any static Capture may be redefined to an MCC in a later Advertisement):

- o Receiving RTP or RTCP containing a CaptureID when the most recently processed Advertisement means that none are expected.
- o Receiving RTP or RTCP without CaptureIDs when the most recently processed Advertisement means that media CaptureIDs are expected.
- o Receiving a CaptureID in RTP or RTCP for a Capture defined in the most recently processed Advertisement, but which the same Advertisement does not include in the MCC.
- o Receiving a CaptureID in RTP or RTCP for a Capture not defined in the most recently processed Advertisement.

7. Multiplexing of CLUE-controlled media using BUNDLE

7.1. 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 separate 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.

7.2. 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.

7.2.1. 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 generallly 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.

7.2.2. 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.

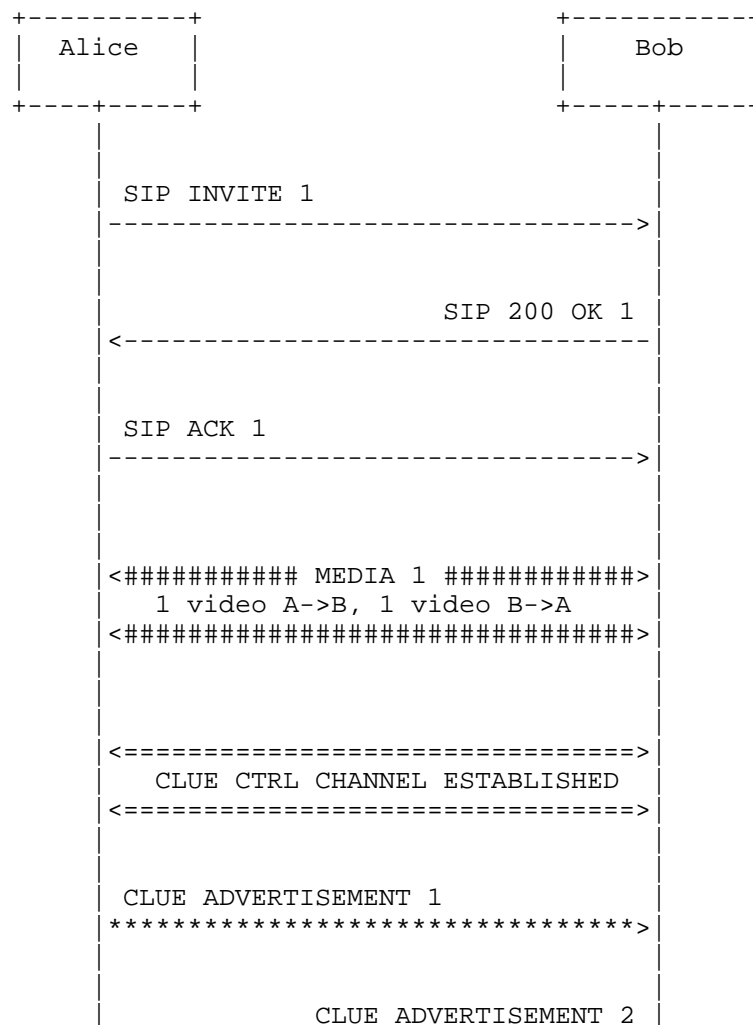
7.2.3. Multiplexing of the data channel and RTP media

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.

8. 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 SDP snippet 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
<-----

```


CLUE negotiation can begin. This is illustrated as CLUE CTRL 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 Captures or the one switched Capture. 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, Alice 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 represent the three CLUE-controlled 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, data channel and the video "m" lines are shown below:


```
...
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=application 6100 UDP/DTLS/SCTP webrtc-datachannel
a=sctp-port: 5000
a=dcmap:2 subprotocol="CLUE";ordered=true
a=mid:3
...
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 ADVERTISEMENT 1.

Alice receives Bob's message CONFIGURE 1 and sends RESPONSE 1 to ack its reception. 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, not shown):

```
...
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, not shown):

```
...
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. Alice also now deactivates the initial non-CLUE-controlled media, as bidirectional CLUE-controlled media is now available. A snippet of

the SDP showing the grouping attribute and the video m-lines are shown below (mid 3 represents the data channel, not shown):

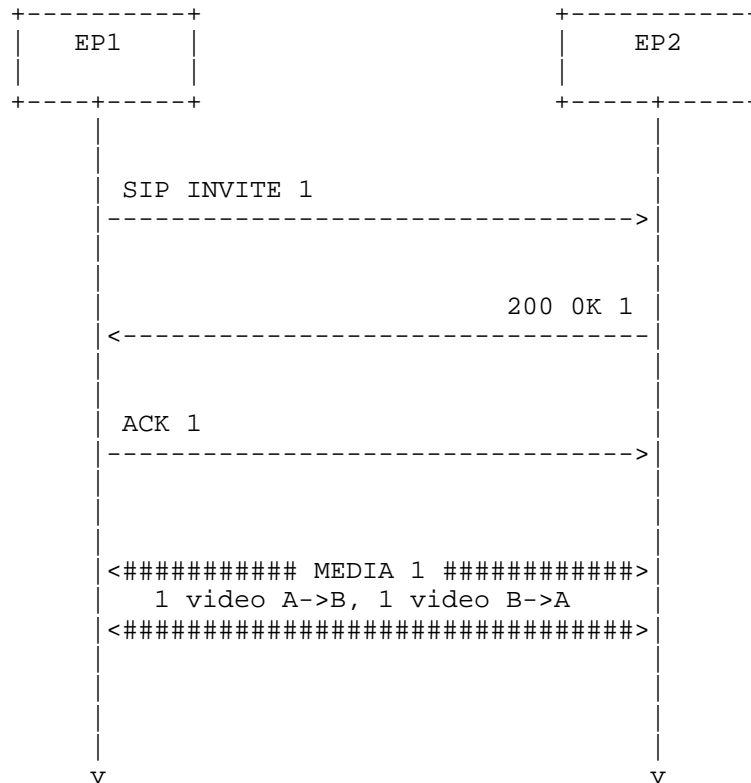
```
...
a=group:CLUE 3 4 5 7 8
...
m=video 0 RTP/AVP 96
a=mid:2
...
m=video 6004 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016
a=sendonly
a=mid:4
a=label:enc1
m=video 6006 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016
a=sendonly
a=mid:5
a=label:enc2
m=video 0 RTP/AVP 96
m=video 6010 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42e016;max-mps=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-mps=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 message or new SDP negotiation to add, remove or change what they have available or want to receive.

9. 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. 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 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 for a DTLS/SCTP channel [I-D.ietf-mmusic-sctp-sdp]. A snippet of the SDP showing the grouping attribute, data channel and the video m-line are shown below:

```

...
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
...
m=application 6100 UDP/DTLS/SCTP webrtc-datachannel
a=sctp-port: 5000
a=dcmap:2 subprotocol="CLUE";ordered=true
a=mid:3

```

Bob is not CLUE-capable, and hence does not recognize the "CLUE" semantic for grouping attribute, nor does he support the data channel. He responds with an answer with audio and video, but with the data channel zeroed.

From the lack of the data channel and grouping framework 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.

10. 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.

11. IANA Considerations

11.1. New SDP Grouping Framework Attribute

This document registers the following semantics with IANA in the "Semantics for the "group" SDP Attribute" subregistry (under the "Session Description Protocol (SDP) Parameters" registry: Semantics Token Reference -----
 ----- CLUE controlled m-line CLUE [this draft]

11.2. New SIP Media Feature Tag

This specification registers a new media feature tag in the SIP [RFC3264] tree per the procedures defined in [RFC2506] and [RFC3840]. Media feature tag name: sip.clue ASN.1 Identifier: 1.3.6.1.8.4.26 Summary of the media feature indicated by this tag: This feature tag indicates that the device supports CLUE controlled media. Values appropriate for use with this feature tag: Boolean. The feature tag is intended primarily for use in the following applications, protocols, services, or negotiation mechanisms: This feature tag is most useful in a communications application for describing the capabilities of a device which uses multiple media streams.

12. Security Considerations

CLUE makes use of a number of protocols and mechanism, either defined by CLUE or long-standing. The security considerations section of the CLUE Framework [I-D.ietf-clue-framework] addresses the need to secure these mechanisms by following the recommendations of the individual protocols.

Beyond the need to secure the constituent protocols, the use of CLUE does impose additional security concerns. One area of increased risk involves the potential for a malicious party to subvert a CLUE-capable device to attack a third party by driving large volumes of media (particularly video) traffic at them by establishing a connection to the CLUE-capable device and directing the media to the victim. While this is a risk for all media devices, a CLUE-capable device may allow the attacker to configure multiple media streams to be sent, significantly increasing the volume of traffic directed at the victim.

This attack can be prevented by ensuring that the media recipient intends to receive the media packets. As such all CLUE-capable devices MUST support key negotiation and receiver intent assurance via DTLS [RFC5763] on CLUE-controlled RTP "m" lines. All CLUE-controlled RTP "m" lines must be secured and implemented using mechanisms such as SRTP [RFC3711]; no specific security mechanisms are made mandatory to use due to the issues addressed in [RFC7202]. Due to the requirements of backwards compatibility, there is not a mandatory requirement for non-CLUE-controlled "m" lines.

CLUE also defines a new media feature tag that indicates CLUE support. This tag may be present even in non-CLUE calls, which increases the metadata available about the sending device, which can help an attacker differentiate between multiple devices and help them identify otherwise anonymised users via the fingerprint of features

their device supports. To prevent this, SIP signalling SHOULD always be encrypted using TLS [RFC5630].

13. Change History

Revision by Rob Hansen

- o State machine interactions updated to match versions in -04 of protocol doc.
- o Section on encoding updated to specify both encID and encodingID from data model doc.
- o Removed the limitations on describing H264 encoding limits using SDP syntax as an open issue.
- o Previous draft had SRTP and DTLS mandatory to implement and to use on CLUE- controlled m lines. Current version has DTLS mandatory to implement, and 'security' mandatory to use but does not define what that security is.
- o Terminology reference to framework doc reinforced. All terminology that duplicates framework removed. All text updated with capitalisation that matches framework document's terminology.
- o SDP example syntax updated to match that of ietf-clue-datachannel and hence ietf-mmusic-data-channel-sdpneg.

Revision by Rob Hansen

- o SRTP/DTLS made mandatory for CLUE-controlled media lines.
- o IANA consideration section added (text as proposed by Christian Groves).
- o Includes provision for dependent streams on seperate "m" lines having the same encID as their parent "m" line.
- o References to putting CLUE-controlled media and data channels in more than one CLUE group removed, since the document no longer supports using more than one CLUE group.
- o Section on CLUE controlled media restrictions still applying even if the call does not end up being CLUE enabled being rewritten to hopefully be clearer.
- o Other minor syntax improvements.

Revision by Rob Hansen

- o Updated DTLS/SCTP channel syntax in examples to fix errors and match latest format defined in draft-ietf-mmusic-sctp-sdp-07.
- o Clarified the behaviour if an SDP offer includes a CLUE-controlled "m" line and the answer accepts that "m" line but without CLUE control of that line.
- o Added a new section on the sending and receiving of CaptureIDs in RTP and RTCP. Includes a section on the necessity of the receiver coping with unexpected CaptureIDs (or the lack thereof) due to MCCs being redefined in new Advertisement messages.
- o Added reminder on IANA section on registering grouping semantic and media feature tag, removed the less formal sections that did the same job.
- o Fixed and clarified issues raised by Christian's document review.
- o Added a number of security considerations.

Revision by Rob Hansen

- o Clarified text on not rejecting messages because they contain unknown encIDs.
- o Removed normative language in section on accepting/rejecting non-CLUE-controlled media in the initial answer.
- o Example SDP updated to include the data channel "m" lines.
- o Example call flow updated to show disablement of non-CLUE-controlled media once CLUE-controlled media is flowing.

-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 Advertisement 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, separate 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 individual 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.ietf-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.

14. References

14.1. Normative References

[I-D.ietf-clue-framework]

Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-22 (work in progress), April 2015.

[I-D.ietf-clue-data-model-schema]

Presta, R. and S. Romano, "An XML Schema for the CLUE data model", draft-ietf-clue-data-model-schema-10 (work in progress), June 2015.

[I-D.ietf-clue-protocol]

Presta, R. and S. Romano, "CLUE protocol", draft-ietf-clue-protocol-04 (work in progress), April 2015.

[I-D.ietf-clue-datachannel]

Holmberg, C., "CLUE Protocol data channel", draft-ietf-clue-datachannel-09 (work in progress), March 2015.

- [I-D.ietf-clue-rtp-mapping]
Even, R. and J. Lennox, "Mapping RTP streams to CLUE media captures", draft-ietf-clue-rtp-mapping-04 (work in progress), March 2015.
- [I-D.ietf-mmusic-sctp-sdp]
Holmberg, C., Loreto, S., and G. Camarillo, "Stream Control Transmission Protocol (SCTP)-Based Media Transport in the Session Description Protocol (SDP)", draft-ietf-mmusic-sctp-sdp-14 (work in progress), March 2015.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <<http://www.rfc-editor.org/info/rfc3711>>.
- [RFC4574] Levin, O. and G. Camarillo, "The Session Description Protocol (SDP) Label Attribute", RFC 4574, DOI 10.17487/RFC4574, August 2006, <<http://www.rfc-editor.org/info/rfc4574>>.
- [RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, DOI 10.17487/RFC5763, May 2010, <<http://www.rfc-editor.org/info/rfc5763>>.

14.2. Informative References

- [RFC2506] Holtman, K., Mutz, A., and T. Hardie, "Media Feature Tag Registration Procedure", BCP 31, RFC 2506, DOI 10.17487/RFC2506, March 1999, <<http://www.rfc-editor.org/info/rfc2506>>.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <<http://www.rfc-editor.org/info/rfc3264>>.
- [RFC3311] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, DOI 10.17487/RFC3311, October 2002, <<http://www.rfc-editor.org/info/rfc3311>>.

- [RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, DOI 10.17487/RFC3840, August 2004, <<http://www.rfc-editor.org/info/rfc3840>>.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <<http://www.rfc-editor.org/info/rfc5245>>.
- [RFC5630] Audet, F., "The Use of the SIPS URI Scheme in the Session Initiation Protocol (SIP)", RFC 5630, DOI 10.17487/RFC5630, October 2009, <<http://www.rfc-editor.org/info/rfc5630>>.
- [RFC6184] Wang, Y., Even, R., Kristensen, T., and R. Jesup, "RTP Payload Format for H.264 Video", RFC 6184, DOI 10.17487/RFC6184, May 2011, <<http://www.rfc-editor.org/info/rfc6184>>.
- [RFC7202] Perkins, C. and M. Westerlund, "Securing the RTP Framework: Why RTP Does Not Mandate a Single Media Security Solution", RFC 7202, DOI 10.17487/RFC7202, April 2014, <<http://www.rfc-editor.org/info/rfc7202>>.
- [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-23 (work in progress), July 2015.

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