

CLUE WG
Internet-Draft
Intended status: Informational
Expires: December 24, 2014

R. Even, Ed.
Huawei Technologies
June 22, 2014

CLUE protocol Call Flows
draft-even-clue-call-flows-00.txt

Abstract

This document provides some call flows examples using the CLUE extensions for "telepresence"

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 December 24, 2014.

Copyright Notice

Copyright (c) 2014 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	2
3. Symmetric point to point Telepresence call	2
4. Acknowledgements	8
5. IANA Considerations	8
6. Security Considerations	8
7. Informative References	9
Author's Address	9

1. Introduction

This document provides some call flows examples using the CLUE extensions for "telepresence". The examples include a typical point to point call between two endpoint with three cameras and screens. A call from a telepresence endpoint to an endpoint that do not support the CLUE telepresence extensions. An point to point call between a three screens and three camera endpoint to a single screen and single camera end point both support the CLUE telepresence extensions.

The examples will not include ICE and SRTP negotiations but the actual usage SHOULD include ICE and SRTP.

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. Symmetric point to point Telepresence call

In this example both end points have three monitors and three cameras and fully support the CLUE telepresence extensions.

The initial call is from Alice to Bob. The first offer includes an audio and video channel, a data channel for CLUE and the CLUE feature tag.

```
INVITE sip:bob@biloxi.example.com SIP/2.0
```

```
Via: SIP/2.0/TCP  
client.atlanta.example.com:5060;branch=z9hG4bK74bf9
```

```
Max-Forwards: 70
```

```
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
```

Call-ID: 3848276298220188511@atlanta.example.com

CSeq: 1 INVITE

Contact: sip:alice@client.atlanta.example.com;transport=tcp; CLUE
(?)

Content-Type: application/sdp

Content-Length: xxx

v=0

o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com

s=-

c=IN IP4 192.0.2.101

t=0 0

m=audio 49172 RTP/AVP 0

a=rtpmap:0 PCMU/8000

m=video 49174 RTP/AVP 96

a=rtpmap:96 H264/90000

a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600

a=sendrecv

m=application 54111 DTLS/SCTP 54111

a=sctpmap:54111 webrtc-datachannel

SIP/2.0 200 OK

Via: SIP/2.0/TCP
client.atlanta.example.com:5060;branch=z9hG4bK74bf9

;received=192.0.2.101

From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl

To: Bob <sip:bob@biloxi.example.com>;tag=8321234356

Call-ID: 3848276298220188511@atlanta.example.com

CSeq: 1 INVITE

Contact: sip:bob@client.biloxi.example.com;transport=tcp; CLUE (?)

Content-Type: application/sdp

Content-Length: zzz

v=0

o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com

s=-

c=IN IP4 192.0.2.201

t=0 0

m=audio 3456 RTP/AVP 0

a=rtpmap:0 PCMU/8000

m=video 3458 RTP/AVP 96

a=rtpmap:96 H264/90000

a=fmtp:96 profile-level-id=42e016;max-mbps=108000;max-fs=3600

a=sendrecv

m=application 54100 DTLS/SCTP 54111

a=sctpmap:54100 webrtc-datachannel

ACK sip:bob@client.biloxi.example.com SIP/2.0

Via: SIP/2.0/TCP
client.atlanta.example.com:5060;branch=z9hG4bK74bd5

Max-Forwards: 70

From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl

To: Bob <sip:bob@biloxi.example.com>;tag=8321234356

Call-ID: 3848276298220188511@atlanta.example.com

CSeq: 1 ACK

Content-Length: 0

After establishing the initial SIP connection Alice and Bob need to open the CLUE channel.

The CLUE data channel is based on the RTCweb data channel as specified in <http://tools.ietf.org/html/draft-ietf-clue-datachannel-00>.

The first step is to open the DTLS [RFC6347] connection . The DTLS connection will be opened by Alice

Alice		Bob
ClientHello	----->	
		ServerHello
		Certificate*
		ServerKeyExchange*
		CertificateRequest*
	<-----	ServerHelloDone
Certificate*		
ClientKeyExchange		
CertificateVerify*		
[ChangeCipherSpec]		
Finished	----->	
		[ChangeCipherSpec]
	<-----	Finished
Application Data	<----->	Application Data

After establishing the DTLS connection the SCTP association is created as specified in [RFC4960]. The INIT and INITACK include the number of channels that will be used.

```

Alice (A)                                     Bob (Z)

INIT [I-Tag=Tag_A OS=1 MIS=1
      I-TSN=0 & other info] ----->
                                     INIT ACK [Veri Tag=Tag_A,
                                                I-Tag=Tag_Z,
<----- Cookie_Z, & other info]

COOKIE ECHO [Cookie_Z] ----->
                                     <----- COOKIE-ACK

```

The SCTP messages are carried in the DATA messages.

The next step is to open a web RTC channel [I-D.ietf-rtcweb-data-protocol]. PPID 50 is the webRTC Data Channel Establishment Protocol (DCEP) [I-D.ietf-rtcweb-data-protocol]]. PPID 51 is the CLUE protocol ID [I-D.ietf-clue-datachannel].

The SCTP DATA message is as follows, the Stream Sequence number will progress.

```

DATA [TSN=initial TSN=0
      Strm=0,Seq=0
      ppid= 50; & user data]----->
                                     SACK [TSN = 0,
<----- Block=0]

```

The first SCTP data message from Alice will carry the DATA_CHANNEL_OPEN message. This will open a bi-directional channel. DATA_CHANNEL_OPEN [message type=3, DATA_CHANNEL_RELIABLE, Label Length = 0, Protocol Length = 4, protocol=CLUE)

Bob Answers with DATA_CHANNEL_ACK [message type=2]

The next SCTP DATA messages will use PPID = 51 since it will carry the CLUE protocol. The Clue Exchange starts from Alice

Question: do we want full XML for CLUE messages or just pseudo code providing the parameters?

```
Alice                                Bob
Option [sequenceNr=1,
media provider=true,
media consumer=true]. ----->
                                OptionResponse
                                [sequenceNr=4
                                ResponseCode,
                                ResponseString,
                                media provider=true,
                                <----- media consumer=true].
```

Alice sends an advertisement to Bob, Alice will also send a new SIP invite with the sendonly CLUE media streams. The SIP call flow is in section 7 of [I-D.ietf-clue-signaling] (should it be moved here?)

```
Advertisement [sequenceNr =2,
mediacaptures,
encodinggroups,
captureScenes] ----->
```

Bob can now send a Configure message asking for the three cameras and video, a SIP message that will specify receive only RTP streams for the m-lines in the offer from ALICE with sendonly streams . The advertisement acknowledge to Alice is in the configure message.

Bob will also send an Advertisement and a SIP INVITE with the send only RTP media streams.

```

Configure [sequenceNr=6,
          advsequenceNr=2
          ack=true
          <----- captureEncodings]
Configure Response [sequenceNr=3,
                  ResponseCode,
                  ResponseString,
                  confSequenceNr=6]----->

```

```

Advertisement
[sequenceNr =7,
mediacaptures,
encodinggroups,
<----- captureScenes]

```

Alice will now send the CONFIGURE message and the SIP Invite for receiving the send only RTP streams from Bob

```

Configure [sequenceNr=4,
advsequenceNr=7
ack=true
captureEncodings] ----->
Configure Response
[sequenceNr=8,
ResponseCode,
ResponseString,
<----- confSequenceNr=4]

```

4. Acknowledgements

5. IANA Considerations

This document contains no IANA considerations.

6. Security Considerations

While there are likely to be security considerations for any solution for telepresence , this document has no security considerations.

7. Informative 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-05 (work in progress), June 2014.
- [I-D.ietf-clue-datachannel]
Holmberg, C., "CLUE Protocol Data Channel", draft-ietf-clue-datachannel-00 (work in progress), March 2014.
- [I-D.ietf-clue-protocol]
Presta, R. and S. Romano, "CLUE protocol", draft-ietf-clue-protocol-00 (work in progress), June 2014.
- [I-D.ietf-clue-signaling]
Kyzivat, P., Xiao, L., Groves, C., and R. Hansen, "CLUE Signaling", draft-ietf-clue-signaling-01 (work in progress), May 2014.
- [I-D.ietf-rtcweb-data-protocol]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel Establishment Protocol", draft-ietf-rtcweb-data-protocol-06 (work in progress), June 2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3265] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

Author's Address

Roni Even (editor)
Huawei Technologies
Tel Aviv
Israel

Email: roni.even@mail01.huawei.com

CLUE Working Group
Internet-Draft
Intended status: Informational
Expires: December 29, 2014

R. Presta
S P. Romano
University of Napoli
June 27, 2014

An XML Schema for the CLUE data model
draft-ietf-clue-data-model-schema-06

Abstract

This document provides an XML schema file for the definition of CLUE data model types.

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 December 29, 2014.

Copyright Notice

Copyright (c) 2014 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. XML Schema	7
4. <mediaCaptures>	17
5. <encodingGroups>	17
6. <captureScenes>	17
7. <simultaneousSets>	17
8. <globalSceneEntries>	17
9. <captureEncodings>	17
10. <mediaCapture>	18
10.1. <capturedMedia>	19
10.2. <captureSceneIDREF>	20
10.3. <encGroupIDREF>	20
10.4. <spatialInformation>	20
10.4.1. <capturePoint>	21
10.4.2. <captureArea>	22
10.5. <nonSpatiallyDefinable>	23
10.6. <content>	23
10.7. <synchronizationID>	24
10.8. <policy>	24
10.9. <maxCaptures>	24
10.10. <individual>	25
10.11. <description>	25
10.12. <priority>	25
10.13. <lang>	26
10.14. <mobility>	26
10.15. <maxCaptureEncodings>	26
10.16. <relatedTo>	26
10.17. <view>	26
10.18. <presentation>	26
10.19. <capturedPeople>	27
10.19.1. <personIDREF>	27
10.20. captureID attribute	27
11. Audio captures	27
11.1. <audioChannelFormat>	27
12. Video captures	28
12.1. <embeddedText>	28
13. Text captures	29
14. <captureScene>	29
14.1. <sceneInformation>	30
14.2. <sceneEntries>	30
14.3. sceneID attribute	31
14.4. scale attribute	31
15. <sceneEntry>	31
15.1. <mediaCaptureIDs>	32
15.2. sceneEntryID attribute	32
15.3. mediaType attribute	32
16. <encodingGroup>	33

16.1.	<maxGroupBandwidth>	33
16.2.	<encodingIDList>	33
16.3.	encodingGroupID attribute	34
17.	<simultaneousSet>	34
17.1.	<captureIDREF>	34
17.2.	<sceneEntryIDREF>	34
18.	<globalSceneEntry>	34
19.	<people>	35
19.1.	<person>	36
19.1.1.	personID attribute	37
19.1.2.	<personInfo>	37
19.1.3.	<personType>	37
20.	<captureEncoding>	37
20.1.	<captureID>	38
20.2.	<encodingID>	38
20.3.	<configuredContent>	38
21.	<clueInfo>	38
22.	XML Schema extensibility	39
23.	Sample XML file	39
24.	MCC example	45
25.	Diff with draft-ietf-clue-data-model-schema-02 version	52
26.	Diff with draft-ietf-clue-data-model-schema-03 version	53
27.	Diff with draft-ietf-clue-data-model-schema-04 version	53
28.	Informative References	54

1. Introduction

This document provides an XML schema file for the definition of CLUE data model types.

The schema is based on information contained in [I-D.ietf-clue-framework]. It encodes information and constraints defined in the aforementioned document in order to provide a formal representation of the concepts therein presented. The schema definition is intended to be modified according to changes applied to the above mentioned CLUE document.

The document aims at the definition of a coherent structure for all the information associated with the description of a telepresence scenario. Such information is used within the CLUE protocol messages ([I-D.ietf-clue-protocol]) enabling the dialogue between a Media Provider and a Media Consumer. CLUE protocol messages, indeed, are XML messages allowing (i) a Media Provider to advertise its telepresence capabilities in terms of media captures, capture scenes, and other features envisioned in the CLUE framework, according to the format herein defined and (ii) a Media Consumer to request the desired telepresence options in the form of capture encodings, represented as described in this document.

2. Terminology

This document refers to the same terminology used in [I-D.ietf-clue-framework], except for the "CLUE Participant" definition (which is still under discussion). We briefly recall herein some of the main terms exploited in the document.

Audio Capture: Media Capture for audio. Denoted as AC_n in the example cases in this document.

Camera-Left and Right: For Media Captures, camera-left and cameraright are from the point of view of a person observing the rendered media. They are the opposite of Stage-Left and Stage-Right.

Capture: Same as Media Capture.

Capture Device: A device that converts audio and video input into an electrical signal, in most cases to be fed into a media encoder.

Capture Encoding: A specific encoding of a Media Capture, to be sent by a Media Provider to a Media Consumer via RTP.

Capture Scene: An abstraction grouping semantically-coupled Media Captures available at the Media Provider's side, representing a precise portion of the local scene that can be transmitted remotely. Capture Scene MAY correspond to a part of the telepresence room or MAY focus only on the presentation media. A Capture Scene is characterized by a set of attributes and by a set of Capture Scene Entries.

Capture Scene Entry: A list of Media Captures of the same media type that constitute a possible representation of a Capture Scene. Media Capture belonging to the same Capture Scene Entry can be sent simultaneously by the Media Provider.

CLUE Participant: This term is not imported from the framework terminology and should be considered temporary since it is under review. We introduced it for the sake of simplicity in order to identify a generic entity (either an Endpoint or a MCU) making use of the CLUE protocol.

Consumer: Same as Media Consumer.

Encoding or Individual Encoding: The representation of an encoding technology. In the CLUE datamodel, for each encoding it is provided a set of parameters representing the encoding constraints, like for example the maximum bandwidth of the Media Provider the encoding can consume. s

Encoding Group: The representation of a group of encodings. For each group, it is provided a set of parameters representing the constraints to be applied to the group as a whole. An example is the maximum bandwidth that can be consumed when using the contained encodings together simultaneously.

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 SIP Conferencing Framework 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. An MCU may include a Mixer.

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 of a single type (i.e., audio or video or text).

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 a MCU) able to send Media Streams.

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

Scene: Same as Capture Scene.

Scene Entry: Same as Capture Scene Entry.

Stream: Same of Media Stream.

Multiple Content Capture: A Capture that can contain different Media Captures of the same media type. It is denoted as MCC in this document. In the Stream resulting from the MCC, the Stream coming from the encoding of the composing Media Captures can appear simultaneously, if the MCC is the result of a mixing operation, or can appear alternatively over the time, according to a certain switching policy.

Plane of Interest: The spatial plane containing the most relevant subject matter.

Provider: Same as Media Provider.

Render:

Simultaneous Transmission Set: a set of Media Captures that can be transmitted simultaneously from a Media Provider.

Single Media Capture: A Capture representing the Media coming from a single-source Capture Device.

Spatial Information: Data about the spatial position of a Capture Device that generate a Single Media Capture within the context of a Capture Scene representing a phisical portion of a Telepresence Room.

Stream Characteristics: The union of the features used to describe a Stream in the CLUE environment and in the SIP-SDP environment.

Video Capture: A Media Capture for video.

3. XML Schema

This section contains the CLUE data model schema definition.

The element and attribute definitions are formal representation of the concepts needed to describe the capabilities of a Media Provider and the streams that are requested by a Media Consumer given the Media Provider's ADVERTISEMENT ([I-D.ietf-clue-protocol]).

The main groups of information are:

<mediaCaptures>: the list of media captures available (Section 4)

<encodingGroups>: the list of encodings groups (Section 5)

<captureScenes>: the list of capture scenes (Section 6)

<simultaneousSets>: the list of simultaneous transmission sets (Section 7)

<globalSceneEntries>: the list of global capture entries sets (Section 8)

<people>: meta data about the participants represented in the telepresence session (Section 19).

<captureEncodings>: the list of instantiated capture encodings (Section 9)

All of the above refers to concepts that have been introduced in [I-D.ietf-clue-framework] and further detailed in the following of this document.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
  targetNamespace="urn:ietf:params:xml:ns:clue-info"
  xmlns:tns="urn:ietf:params:xml:ns:clue-info"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:ietf:params:xml:ns:clue-info"
  xmlns:xcard="urn:ietf:params:xml:ns:vcard-4.0"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
```



```
<!-- Import xcard XML schema -->
<xs:import namespace="urn:ietf:params:xml:ns:vcard-4.0"
schemaLocation="xcard.xsd"/>

<!-- ELEMENT DEFINITIONS -->
<xs:element name="mediaCaptures" type="mediaCapturesType"/>
<xs:element name="encodingGroups" type="encodingGroupsType"/>
<xs:element name="captureScenes" type="captureScenesType"/>
<xs:element name="simultaneousSets" type="simultaneousSetsType"/>
<xs:element name="globalSceneEntries" type="globalSceneEntriesType"/>
<xs:element name="people" type="peopleType"/>

<xs:element name="captureEncodings" type="captureEncodingsType"/>

<!-- MEDIA CAPTURES TYPE -->
<!-- envelope of media captures -->
<xs:complexType name="mediaCapturesType">
  <xs:sequence>
    <xs:element name="mediaCapture" type="mediaCaptureType"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- DESCRIPTION element -->
<xs:element name="description">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="lang" type="xs:language"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<!-- MEDIA CAPTURE TYPE -->
<xs:complexType name="mediaCaptureType" abstract="true">
  <xs:sequence>
    <!-- mandatory fields -->
    <xs:element name="capturedMedia" type="xs:string"/>
    <xs:element name="captureSceneIDREF" type="xs:IDREF"/>
    <xs:element name="encGroupIDREF" type="xs:IDREF" minOccurs="0"/>
    <xs:choice>
      <xs:sequence>
        <xs:element name="spatialInformation"
          type="tns:spatialInformationType"/>
      </xs:sequence>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

```

    </xs:sequence>
    <xs:element name="nonSpatiallyDefinable" type="xs:boolean" fixed="true"/>
  </xs:choice>
  <!-- for handling multi-content captures: -->
  <xs:choice>
    <xs:sequence>
      <xs:element name="synchronizationID" type="xs:ID" minOccurs="0"/>
      <xs:element name="content" type="contentType" minOccurs="0"/>
      <xs:element name="policy" type="xs:string" minOccurs="0"/>
      <xs:element name="maxCaptures" type="maxCapturesType" minOccurs="0"/>
    </xs:sequence>
    <xs:element name="individual" type="xs:boolean" fixed="true"/>
  </xs:choice>
  <!-- optional fields -->
  <xs:element ref="description" minOccurs="0" maxOccurs="unbounded"/>
  <xs:element name="priority" type="xs:unsignedInt" minOccurs="0"/>
  <xs:element name="lang" type="xs:language" minOccurs="0"/>
  <xs:element name="mobility" type="mobilityType" minOccurs="0"/>
  <xs:element name="presentation" type="presentationType" minOccurs="0"/>
  <xs:element name="view" type="viewType" minOccurs="0"/>
  <xs:element name="capturedPeople" type="capturedPeopleType" minOccurs="0"/>
  <xs:element name="maxCaptureEncodings" type="xs:unsignedInt"
minOccurs="0"/>
  <xs:element name="relatedTo" type="xs:IDREF" minOccurs="0"/>
  <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="captureID" type="xs:ID" use="required"/>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- CONTENT TYPE -->
<xs:complexType name="contentType">
  <xs:sequence>
    <xs:element name="captureIDREF" type="xs:IDREF"
minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="sceneEntryIDREF" type="xs:IDREF"
minOccurs="0" maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- MAX CAPTURES TYPE -->
<xs:complexType name="maxCapturesType">
  <xs:simpleContent>
```

```
<xs:extension base="xs:unsignedInt">
  <xs:attribute name="exactNumber" type="xs:boolean"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>

<!-- CAPTURED PEOPLE TYPE -->
<xs:complexType name="capturedPeopleType">
<xs:sequence>
<xs:element name="personIDREF" type="xs:IDREF" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<!-- PEOPLE TYPE -->
<xs:complexType name="peopleType">
<xs:sequence>
<xs:element name="person" type="personType"
              maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<!-- PERSON TYPE -->
<xs:complexType name="personType">
  <xs:sequence>
    <xs:element name="personInfo" type="xcard:vcardType" maxOccurs="1"
                  minOccurs="0"/>
    <xs:element name="personType" type="personTypeType"
                  minOccurs="0"
                  maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
              maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="personID" type="xs:ID" use="required"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- PERSON TYPE TYPE -->
<xs:simpleType name="personTypeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="chairman"/>
    <xs:enumeration value="vice-chairman"/>
    <xs:enumeration value="minute taker"/>
    <xs:enumeration value="presenter"/>
    <xs:enumeration value="translator"/>
    <xs:enumeration value="timekeeper"/>
    <xs:enumeration value="attendee"/>
  </xs:restriction>
</xs:simpleType>
```

```
<!-- VIEW TYPE -->
<xs:simpleType name="viewType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="room"/>
    <xs:enumeration value="table"/>
    <xs:enumeration value="lectern"/>
    <xs:enumeration value="individual"/>
    <xs:enumeration value="audience"/>
  </xs:restriction>
</xs:simpleType>

<!-- PRESENTATION TYPE -->
<xs:simpleType name="presentationType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="slides"/>
    <xs:enumeration value="image"/>
    <xs:enumeration value=""/>
  </xs:restriction>
</xs:simpleType>

<!-- SPATIAL INFORMATION TYPE -->
<xs:complexType name="spatialInformationType">
  <xs:sequence>
    <xs:element name="capturePoint" type="capturePointType"/>
    <xs:element name="captureArea" type="captureAreaType"
      minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- MOBILITY TYPE -->
<xs:simpleType name="mobilityType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="static"/>
    <xs:enumeration value="dynamic"/>
    <xs:enumeration value="highly-dynamic"/>
  </xs:restriction>
</xs:simpleType>

<!-- TEXT CAPTURE TYPE -->
<xs:complexType name="textCaptureType">
  <xs:complexContent>
    <xs:extension base="tns:mediaCaptureType">
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
<!-- AUDIO CAPTURE TYPE -->
<xs:complexType name="audioCaptureType">
  <xs:complexContent>
    <xs:extension base="tns:mediaCaptureType">
      <xs:sequence>
        <xs:element name="audioChannelFormat" type="audioChannelFormatType"
          minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
<!-- AUDIO CHANNEL FORMAT TYPE -->
<xs:simpleType name="audioChannelFormatType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="mono"/>
    <xs:enumeration value="stereo"/>
  </xs:restriction>
</xs:simpleType>
```

```
<!-- VIDEO CAPTURE TYPE -->
<xs:complexType name="videoCaptureType">
  <xs:complexContent>
    <xs:extension base="tns:mediaCaptureType">
      <xs:sequence>
        <xs:element ref="embeddedText" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
<!-- EMBEDDED TEXT ELEMENT -->
<xs:element name="embeddedText">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:boolean">
        <xs:attribute name="lang" type="xs:language"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

```
<!-- CAPTURE SCENES TYPE -->
<!-- envelope of capture scenes -->
<xs:complexType name="captureScenesType">
  <xs:sequence>
    <xs:element name="captureScene" type="captureSceneType">
```

```
    maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- CAPTURE SCENE TYPE -->
<xs:complexType name="captureSceneType">
  <xs:sequence>
    <xs:element ref="description" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="sceneInformation" type="xcard:vcardType" minOccurs="0"/>
    <xs:element name="sceneEntries" type="sceneEntriesType"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="sceneID" type="xs:ID" use="required"/>
  <xs:attribute name="scale" type="scaleType" use="required"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- SCALE TYPE -->
<xs:simpleType name="scaleType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="millimeters"/>
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="noscale"/>
  </xs:restriction>
</xs:simpleType>

<!-- CAPTURE AREA TYPE -->
<xs:complexType name="captureAreaType">
  <xs:sequence>
    <xs:element name="bottomLeft" type="pointType"/>
    <xs:element name="bottomRight" type="pointType"/>
    <xs:element name="topLeft" type="pointType"/>
    <xs:element name="topRight" type="pointType"/>
  </xs:sequence>
</xs:complexType>

<!-- POINT TYPE -->
<xs:complexType name="pointType">
  <xs:sequence>
    <xs:element name="x" type="xs:decimal"/>
    <xs:element name="y" type="xs:decimal"/>
    <xs:element name="z" type="xs:decimal"/>
  </xs:sequence>
</xs:complexType>

<!-- CAPTURE POINT TYPE -->
<xs:complexType name="capturePointType">
```

```
<xs:complexContent>
  <xs:extension base="pointType">
    <xs:sequence>
      <xs:element name="lineOfCapturePoint" type="tns:pointType"
        minOccurs="0"/>
    </xs:sequence>
  </xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- SCENE ENTRIES TYPE -->
<!-- envelope of scene entries of a capture scene -->
<xs:complexType name="sceneEntriesType">
  <xs:sequence>
    <xs:element name="sceneEntry" type="sceneEntryType"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- SCENE ENTRY TYPE -->
<xs:complexType name="sceneEntryType">
  <xs:sequence>
    <xs:element ref="description" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="mediaCaptureIDs" type="captureIDListType"/>
  </xs:sequence>
  <xs:attribute name="sceneEntryID" type="xs:ID" use="required"/>
</xs:complexType>

<!-- CAPTURE ID LIST TYPE -->
<xs:complexType name="captureIDListType">
  <xs:sequence>
    <xs:element name="captureIDREF" type="xs:IDREF"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- ENCODING GROUPS TYPE -->
<xs:complexType name="encodingGroupsType">
  <xs:sequence>
    <xs:element name="encodingGroup" type="tns:encodingGroupType"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- ENCODING GROUP TYPE -->
<xs:complexType name="encodingGroupType">
```

```
<xs:sequence>
  <xs:element name="maxGroupBandwidth" type="xs:unsignedInt"/>
  <xs:element name="encodingIDList" type="encodingIDListType"/>
  <xs:any namespace="##other" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="encodingGroupID" type="xs:ID" use="required"/>
<xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<!-- ENCODING ID LIST TYPE -->
<xs:complexType name="encodingIDListType">
  <xs:sequence>
    <xs:element name="encID" type="xs:string" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- SIMULTANEOUS SETS TYPE -->
<xs:complexType name="simultaneousSetsType">
  <xs:sequence>
    <xs:element name="simultaneousSet" type="simultaneousSetType"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- SIMULTANEOUS SET TYPE -->
<xs:complexType name="simultaneousSetType">
  <xs:sequence>
    <xs:element name="captureIDREF" type="xs:IDREF"
      minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="sceneEntryIDREF" type="xs:IDREF"
      minOccurs="0" maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="setID" type="xs:ID" use="required"/>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<!-- GLOBAL SCENE ENTRIES TYPE -->
<xs:complexType name="globalSceneEntriesType">
  <xs:sequence>
    <xs:element name="globalSceneEntry" type="globalSceneEntryType"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- GLOBAL SCENE ENTRY TYPE -->
```



```
<xs:complexType name="globalSceneEntryType">
  <xs:sequence>
    <xs:element name="sceneEntryIDREF" type="xs:IDREF"
      maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="globalSceneEntryID" type="xs:ID" use="required"/>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<!-- CAPTURE ENCODINGS TYPE -->
<xs:complexType name="captureEncodingsType">
  <xs:sequence>
    <xs:element name="captureEncoding" type="captureEncodingType"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- CAPTURE ENCODING TYPE -->
<xs:complexType name="captureEncodingType">
  <xs:sequence>
    <xs:element name="captureID" type="xs:string"/>
    <xs:element name="encodingID" type="xs:string"/>
    <xs:element name="configuredContent" type="contentType" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="ID" type="xs:ID" use="required"/>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<!-- CLUE INFO ELEMENT -->
<!-- the <clueInfo> envelope can be seen
      as the ancestor of an <advertisement> envelope -->
<xs:element name="clueInfo" type="clueInfoType"/>

<!-- CLUE INFO TYPE -->
<xs:complexType name="clueInfoType">
  <xs:sequence>
    <xs:element ref="mediaCaptures"/>
    <xs:element ref="encodingGroups"/>
    <xs:element ref="captureScenes"/>
    <xs:element ref="simultaneousSets" minOccurs="0"/>
    <xs:element ref="globalSceneEntries" minOccurs="0"/>
    <xs:element ref="people" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

```
</xs:sequence>
<xs:attribute name="clueInfoID" type="xs:ID" use="required"/>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
</xs:schema>
```

Following sections describe the XML schema in more detail.

4. <mediaCaptures>

<mediaCaptures> represents the list of one ore more media captures available on the Media Provider's side. Each media capture is represented by a <mediaCapture> element (Section 10).

5. <encodingGroups>

<encodingGroups> represents the list of the encoding groups organized on the Media Provider's side. Each encoding group is represented by a <encodingGroup> element (Section 16).

6. <captureScenes>

<captureScenes> represents the list of the capture scenes organized on the Media Provider's side. Each capture scene is represented by a <captureScene> element. (Section 14).

7. <simultaneousSets>

<simultaneousSets> contains the simultaneous sets indicated by the Media Provider. Each simultaneous set is represented by a <simultaneousSet> element. (Section 17).

8. <globalSceneEntries>

<globalSceneEntries> contains a set of alternative representations of all the scenes that are offered by a Media Provider to a Media Consumer. Each alternative is named "global scene entry" and it is represented by a <globalSceneEntry> element. (Section 18).

9. <captureEncodings>

<captureEncodings> is a list of capture encodings. It can represent the list of the desired capture encodings indicated by the Media Consumer or the list of instantiated captures on the provider's side. Each capture encoding is represented by a <captureEncoding> element. (Section 20).

10. <mediaCapture>

According to the CLUE framework, a media capture is the fundamental representation of a media flow that is available on the provider's side. Media captures are characterized (i) by a set of features that are independent from the specific type of medium, and (ii) by a set of features that are media-specific. The features that are common to all media types appear within the media capture type, that has been designed as an abstract complex type. Media-specific captures, such as video captures, audio captures and others, are specialization of that abstract media capture type, as in a typical generalization-specialization hierarchy.

The following is the XML Schema definition of the media capture type:

```
<!-- MEDIA CAPTURE TYPE -->
<xs:complexType name="mediaCaptureType" abstract="true">
  <xs:sequence>
    <!-- mandatory fields -->
    <xs:element name="capturedMedia" type="xs:string"/>
    <xs:element name="captureSceneIDREF" type="xs:IDREF"/>
    <xs:element name="encGroupIDREF" type="xs:IDREF" minOccurs="0"/>
    <xs:choice>
      <xs:sequence>
        <xs:element name="spatialInformation"
          type="tns:spatialInformationType"/>
      </xs:sequence>
      <xs:element name="nonSpatiallyDefinable" type="xs:boolean" fixed="true"/>
    </xs:choice>
    <!-- for handling multi-content captures: -->
    <xs:choice>
      <xs:sequence>
        <xs:element name="synchronizationID" type="xs:ID" minOccurs="0"/>
        <xs:element name="content" type="contentType" minOccurs="0"/>
        <xs:element name="policy" type="xs:string" minOccurs="0"/>
        <xs:element name="maxCaptures" type="maxCapturesType" minOccurs="0"/>
      </xs:sequence>
      <xs:element name="individual" type="xs:boolean" fixed="true"/>
    </xs:choice>
    <!-- optional fields -->
    <xs:element ref="description" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="priority" type="xs:unsignedInt" minOccurs="0"/>
    <xs:element name="lang" type="xs:language" minOccurs="0"/>
    <xs:element name="mobility" type="mobilityType" minOccurs="0"/>
    <xs:element name="presentation" type="presentationType" minOccurs="0"/>
    <xs:element name="view" type="viewType" minOccurs="0"/>
    <xs:element name="capturedPeople" type="capturedPeopleType" minOccurs="0"/>
    <xs:element name="maxCaptureEncodings" type="xs:unsignedInt"
      minOccurs="0"/>
    <xs:element name="relatedTo" type="xs:IDREF" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="captureID" type="xs:ID" use="required"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

10.1. <capturedMedia>

<capturedMedia> is a mandatory field specifying the media type of the capture ("audio", "video", "text",...).

10.2. <captureSceneIDREF>

<captureSceneIDREF> is a mandatory field containing the identifier of the capture scene the media capture is defined in. Indeed, each media capture must be defined within one and only one capture scene. When a media capture is spatially definable, some spatial information is provided along with it in the form of point coordinates (see Section 10.4). Such coordinates refers to the space of coordinates defined for the capture scene containing the capture.

10.3. <encGroupIDREF>

<encGroupIDREF> is a mandatory field containing the identifier of the encoding group the media capture is associated with.

10.4. <spatialInformation>

Media captures are divided into two categories: (i) non spatially definable captures and (ii) spatially definable captures.

Captures are spatially definable when it is possible to provide at least the coordinates of the device position within the telepresence room of origin. Such coordinates are expressed according to the coordinate space of the capture scene the media captures belongs to.

Non spatially definable captures cannot be characterized within the physical space of the telepresence room of origin. Capture of this kind are for example those related to registrations, text captures, DVDs, registered presentation, or external streams, that are played in the telepresence room and transmitted to remote sites. Another example is represented by switched captures: their content, in fact, comes from different devices over the time.

Spatially definable captures represent a part of the telepresence room. The captured part of the telepresence room is described by means of the <spatialInformation> element. Non spatially definable captures do not show in their XML description such element: they are instead characterized by having the <nonSpatiallyDefinable> tag set to "true" (see Section 10.5).

The definition of the spatial information type is the following:

```
<!-- SPATIAL INFORMATION TYPE -->
<xs:complexType name="spatialInformationType">
  <xs:sequence>
    <xs:element name="capturePoint" type="capturePointType"/>
    <xs:element name="captureArea" type="captureAreaType"
      minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

The <capturePoint> contains the coordinates of the capture device that is taking the capture, as well as, optionally, the pointing direction (see Section 10.4.1). It is a mandatory field when the media capture is spatially definable, independently from the media type.

The <captureArea> is an optional field containing four points defining the captured area covered by the capture (see Section 10.4.2).

10.4.1. <capturePoint>

The <capturePoint> element is used to represent the position and the line of capture of a capture device. The XML Schema definition of the <capturePoint> element type is the following:

```
<!-- POINT TYPE -->
<xs:complexType name="pointType">
  <xs:sequence>
    <xs:element name="x" type="xs:decimal"/>
    <xs:element name="y" type="xs:decimal"/>
    <xs:element name="z" type="xs:decimal"/>
  </xs:sequence>
</xs:complexType>

<!-- CAPTURE POINT TYPE -->
<xs:complexType name="capturePointType">
  <xs:complexContent>
    <xs:extension base="pointType">
      <xs:sequence>
        <xs:element name="lineOfCapturePoint" type="tns:pointType"
          minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The point type contains three spatial coordinates (x,y,z) representing a point in the space associated with a certain capture scene.

The capture point type extends the point type, i.e., it is represented by three coordinates identifying the position of the capture device, but can add further information. Such further information is conveyed by the <lineOfCapturePoint>, which is another point-type element representing the "point on line of capture", that gives the pointing direction of the capture device.

The coordinates of the point on line of capture MUST NOT be identical to the capture point coordinates. If the point on line of capture is not specified, no assumptions are made about the pointing direction of the capturing device.

10.4.2. <captureArea>

<captureArea> is an optional element that can be contained within the spatial information associated with a media capture. It represents the spatial area captured by the media capture.

The XML representation of that area is provided through a set of four point-type element, <bottomLeft>, <bottomRight>, <topLeft>, and <topRight>, as it can be seen from the following definition:

```
<!-- CAPTURE AREA TYPE -->
<xs:complexType name="captureAreaType">
  <xs:sequence>
    <xs:element name="bottomLeft" type="pointType"/>
    <xs:element name="bottomRight" type="pointType"/>
    <xs:element name="topLeft" type="pointType"/>
    <xs:element name="topRight" type="pointType"/>
  </xs:sequence>
</xs:complexType>
```

<bottomLeft>, <bottomRight>, <topLeft>, and <topRight> should be coplanar.

By comparing the capture area of different media captures within the same capture scene, a consumer can better determine the spatial relationships between them and render them correctly.

10.5. <nonSpatiallyDefinable>

When media captures are non spatially definable, they are marked with the boolean <nonSpatiallyDefinable> element set to "true" and no <spatialInformation> is provided. Indeed, <nonSpatiallyDefinable> and <spatialInformation> are mutually exclusive tag, according to the <choice> section within the XML Schema definition of the media capture type.

10.6. <content>

A media capture can be (i) an individual media capture or (ii) a multiple content capture (MMCC). A multiple content capture is made by different captures that can be arranged spatially (by a composition operation), or temporally (by a switching operation), or that can result from the orchestration of both the techniques. If a media capture is a MCC, then it MUST show in its XML data model representation the <content> element. It is a mandatory element composed by a list of media capture identifiers ("captureIDREF") and capture scene entry identifiers ("sceneEntryIDREF"), where the last ones are used as shortcuts to refer to multiple capture identifiers.


```
<!-- CONTENT TYPE -->
<xs:complexType name="contentType">
  <xs:sequence>
    <xs:element name="captureIDREF" type="xs:IDREF"
      minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="sceneEntryIDREF" type="xs:IDREF"
      minOccurs="0" maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

10.7. <synchronizationID>

<synchronizationID> is an optional element for multiple content captures that contains a numeric identifier. Multiple content captures marked with the same identifier in the <synchronizationID> contain at each time captures coming from the same endpoint.

10.8. <policy>

<policy> is an optional element that can be used only for multiple content captures. It indicates the criteria applied to build the multiple content capture using the media captures referenced in <contentCaptureIDs>. Such element can assume a list of pre-defined values ([todo]).

10.9. <maxCaptures>

<maxCaptures> is an optional element that can be used only for multiple content captures. It provides information about the number of media captures that can be represented in the multiple content capture at a time. The type definition is provided below.

```
<!-- MAX CAPTURES TYPE -->
<xs:complexType name="maxCapturesType">
  <xs:simpleContent>
    <xs:extension base="xs:unsignedInt">
      <xs:attribute name="exactNumber" type="xs:boolean"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

When the "exactNumber" attribute is set to "1", it means the <maxCaptures> element carries the exact number of the media captures appearing at a time. Otherwise, the number of the represented media captures MUST be considered "<=" of the <maxCaptures> value.

10.10. <individual>

<individual> is a boolean element that MUST be used for single-content captures. Its value is fixed and set to "true". Such element indicates the capture that is being described is not a multiple content capture. Indeed, <individual> and the aforementioned tags related to MCC attributes (from Section 10.6 to Section 10.9) are mutually exclusive, according to the <choice> section within the XML Schema definition of the media capture type.

10.11. <description>

<description> is used to provide optionally human-readable textual information about a media capture. The same element is exploited to describe, besides media captures, capture scenes and capture scene entries, as it is included in their XML representation. A media capture can be described by using multiple <description> elements, each one providing information in a different language. The <description> element definition is the following:

```
<!-- DESCRIPTION element -->
<xs:element name="description">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="lang" type="xs:language"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

As it can be seen, <description> is a string element with an attribute ("lang") indicating the language used in the textual description.

10.12. <priority>

<priority> is an optional unsigned integer field indicating the importance of a media capture according to the Media Provider's perspective. It can be used on the receiver's side to automatically

identify the most relevant contribution from the Media Provider. The higher the importance, the lower the contained value. When media captures are marked with a "0" priority value, it means that they are "not subject to priority".

10.13. <lang>

<lang> is an optional element containing the language used in the capture, if any.

10.14. <mobility>

<mobility> is an optional element indicating whether or not the capture device originating the capture may move during the telepresence session. That optional element can assume one of the three following values: (i) static, (ii) dynamic or (iii) highly dynamic.

10.15. <maxCaptureEncodings>

The optional <maxCaptureEncodings> contains an unsigned integer indicating the maximum number of capture encodings that can be simultaneously active for the media capture. If absent, this parameter defaults to 1. The minimum value for this attribute is 1. The number of simultaneous capture encodings is also limited by the restrictions of the encoding group the media capture refers to by means of the <encGroupIDREF> element.

10.16. <relatedTo>

The optional <relatedTo> element contains the value of the ID attribute of the media capture it refers to. The media capture marked with a <relatedTo> element can be for example the translation of a main media capture in a different language.

10.17. <view>

The <view> element is an optional tag describing what is represented in the spatial area covered by a media capture. The current possible values are: "table", "lectern", "individual", and "audience", as listed in the enumerative view type in the following.

10.18. <presentation>

The <presentation> element is an optional tag used for media captures conveying information about presentations within the telepresence session. The current possible values are "slides" and "images", as listed in the enumerative presentation type in the following.

10.19. <capturedPeople>

This optional element is used to indicate which telepresence session participants are represented within the media captures. For each participant, a <personIDREF> element is provided.

10.19.1. <personIDREF>

<personIDREF> contains the identifier of the represented person. Metadata about the represented participant can be retrieved by accessing the <people> list (Section 19).

10.20. captureID attribute

The "captureID" attribute is a mandatory field containing the identifier of the media capture.

11. Audio captures

Audio captures inherit all the features of a generic media capture and present further audio-specific characteristics. The XML Schema definition of the audio capture type is reported below:

```
<!-- AUDIO CAPTURE TYPE -->
<xs:complexType name="audioCaptureType">
  <xs:complexContent>
    <xs:extension base="tns:mediaCaptureType">
      <xs:sequence>
        <xs:element name="audioChannelFormat" type="audioChannelFormatType"
          minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Audio-specific information about the audio capture is contained in <audioChannelFormat> (Section 11.1).

11.1. <audioChannelFormat>

The optional <audioChannelFormat> element is a field with enumerated values ("mono" and "stereo") which describes the method of encoding used for audio. A value of "mono" means the audio capture has one channel. A value of "stereo" means the audio capture has two audio channels, left and right. A single stereo capture is different from

two mono captures that have a left-right spatial relationship. A stereo capture maps to a single RTP stream, while each mono audio capture maps to a separate RTP stream.

The XML Schema definition of the <audioChannelFormat> element type is provided below:

```
<!-- AUDIO CHANNEL FORMAT TYPE -->
<xs:simpleType name="audioChannelFormatType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="mono"/>
    <xs:enumeration value="stereo"/>
  </xs:restriction>
</xs:simpleType>
```

12. Video captures

Video captures, similarly to audio captures, extend the information of a generic media capture with video-specific features, such as <embeddedText> (Section 12.1).

The XML Schema representation of the video capture type is provided in the following:

```
<!-- VIDEO CAPTURE TYPE -->
<xs:complexType name="videoCaptureType">
  <xs:complexContent>
    <xs:extension base="tns:mediaCaptureType">
      <xs:sequence>
        <xs:element ref="embeddedText" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

12.1. <embeddedText>

The <embeddedText> element is a boolean element indicating that there is text embedded in the video capture. The language used in such embedded textual description is reported in <embeddedText> "lang" attribute.

The XML Schema definition of the <embeddedText> element is:

```
<!-- EMBEDDED TEXT ELEMENT -->
<xs:element name="embeddedText">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:boolean">
        <xs:attribute name="lang" type="xs:language"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

13. Text captures

Also text captures can be described by extending the generic media capture information, similarly to audio captures and video captures.

The XML Schema representation of the text capture type is currently lacking text-specific information, as it can be seen by looking at the definition below:

```
<!-- TEXT CAPTURE TYPE -->
<xs:complexType name="textCaptureType">
  <xs:complexContent>
    <xs:extension base="tns:mediaCaptureType">
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

14. <captureScene>

A Media Provider organizes the available capture in capture scenes in order to help the receiver both in the rendering and in the selection of the group of captures. Capture scenes are made of capture scene entries, that are set of media captures of the same media type. Each capture scene entry represents an alternative to represent completely a capture scene for a fixed media type.

The XML Schema representation of a <captureScene> element is the following:

```
<!-- CAPTURE SCENE TYPE -->
<xs:complexType name="captureSceneType">
  <xs:sequence>
    <xs:element ref="description" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="sceneInformation" type="xcard:vcardType" minOccurs="0"/>
    <xs:element name="sceneEntries" type="sceneEntriesType"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="sceneID" type="xs:ID" use="required"/>
  <xs:attribute name="scale" type="scaleType" use="required"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

The <captureScene> element can contain zero or more textual <description> elements, defined as in Section 10.11. Besides <description>, there is the <sceneInformation> element (Section 14.1), which contains structured information about the scene in the vcard format, and the <sceneEntries> element (Section 14.2), which is the list of the capture scene entries.

14.1. <sceneInformation>

The <sceneInformation> element contains optional information about the capture scene according to the vcard format.

14.2. <sceneEntries>

The <sceneEntries> element is a mandatory field of a capture scene containing the list of scene entries. Each scene entry is represented by a <sceneEntry> element (Section 15).

```
<!-- SCENE ENTRIES TYPE -->
<!-- envelope of scene entries of a capture scene -->
<xs:complexType name="sceneEntriesType">
  <xs:sequence>
    <xs:element name="sceneEntry" type="sceneEntryType"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

14.3. sceneID attribute

The sceneID attribute is a mandatory attribute containing the identifier of the capture scene.

14.4. scale attribute

The scale attribute is a mandatory attribute that specifies the scale of the coordinates provided in the spatial information of the media capture belonging to the considered capture scene. The scale attribute can assume three different values:

"millimeters" - the scale is in millimeters. Systems which know their physical dimensions (for example professionally installed telepresence room systems) should always provide those real-world measurements.

"unknown" - the scale is not necessarily millimeters, but the scale is the same for every media capture in the capture scene. Systems which don't know specific physical dimensions but still know relative distances should select "unknown" in the scale attribute of the capture scene to be described.

"noscale" - there is no a common physical scale among the media captures of the capture scene. That means the scale could be different for each media capture.

```
<!-- SCALE TYPE -->
<xs:simpleType name="scaleType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="millimeters"/>
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="noscale"/>
  </xs:restriction>
</xs:simpleType>
```

15. <sceneEntry>

A <sceneEntry> element represents a capture scene entry, which contains a set of media capture of the same media type describing a capture scene.

A <sceneEntry> element is characterized as follows.


```
<!-- SCENE ENTRY TYPE -->
<xs:complexType name="sceneEntryType">
  <xs:sequence>
    <xs:element ref="description" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="mediaCaptureIDs" type="captureIDListType"/>
  </xs:sequence>
  <xs:attribute name="sceneEntryID" type="xs:ID" use="required"/>
</xs:complexType>
```

One or more optional <description> elements provide human-readable information about what the scene entry contains. <description> is defined as already seen in Section 10.11.

The remaining child elements are described in the following subsections.

15.1. <mediaCaptureIDs>

The <mediaCaptureIDs> is the list of the identifiers of the media captures included in the scene entry. It is an element of the captureIDListType type, which is defined as a sequence of <captureIDREF> each one containing the identifier of a media capture listed within the <mediaCaptures> element:

```
<!-- CAPTURE ID LIST TYPE -->
<xs:complexType name="captureIDListType">
  <xs:sequence>
    <xs:element name="captureIDREF" type="xs:IDREF"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

15.2. sceneEntryID attribute

The sceneEntryID attribute is a mandatory attribute containing the identifier of the capture scene entry represented by the <sceneEntry> element.

15.3. mediaType attribute

The mediaType attribute contains the media type of the media captures included in the scene entry.

16. <encodingGroup>

The <encodingGroup> element represents an encoding group, which is made by a set of one or more individual encodings and some parameters that apply to the group as a whole. Encoding groups contain references to individual encodings that can be applied to captures of the same media type. In other words, they can group audio encodings or, alternatively, video encodings. The definition of the <encodingGroup> element is the following:

```
<!-- ENCODING GROUP TYPE -->
<xs:complexType name="encodingGroupType">
  <xs:sequence>
    <xs:element name="maxGroupBandwidth" type="xs:integer"/>
    <xs:element name="encodingIDList" type="encodingIDListType"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="encodingGroupID" type="xs:ID" use="required"/>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
```

In the following, the contained elements are further described.

16.1. <maxGroupBandwidth>

<maxGroupBandwidth> is an optional field containing the maximum bitrate that can be shared by the individual encodings included in the encoding group.

16.2. <encodingIDList>

<encodingIDList> is the list of the individual encoding grouped together in the encoding group. Each individual encoding is represented through its identifier contained within an <encID> element.

```
<!-- ENCODING ID LIST TYPE -->
<xs:complexType name="encodingIDListType">
  <xs:sequence>
    <xs:element name="encID" type="xs:IDREF" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

16.3. encodingGroupID attribute

The encodingGroupID attribute contains the identifier of the encoding group.

17. <simultaneousSet>

<simultaneousSet> represents a simultaneous transmission set, i.e. a list of captures of the same media type that can be transmitted at the same time by a Media Provider. There are different simultaneous transmission sets for each media type.

```
<!-- SIMULTANEOUS SET TYPE -->
<xs:complexType name="simultaneousSetType">
  <xs:sequence>
    <xs:element name="captureIDREF" type="xs:IDREF"
      minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="sceneEntryIDREF" type="xs:IDREF"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Besides the identifiers of the captures (<captureIDREF> elements), also the identifiers of capture scene entries can be exploited, as shortcuts (<sceneEntryIDREF> elements).

17.1. <captureIDREF>

<captureIDREF> contains the identifier of the media capture that belongs to the simultaneous set.

17.2. <sceneEntryIDREF>

<captureIDREF> contains the identifier of the scene entry containing a group of capture that are able to be sent simultaneously with the other capture of the simultaneous set.

18. <globalSceneEntry>

<globalSceneEntry> represents a set of captures of the same media type representing a summary of the complete Media Provider's offer. The media type of a global scene entry is reported in the "mediaType" attribute. The content of a global scene entry is expressed by leveraging only scene entry identifiers, put within <sceneEntryIDREF> elements. Each global scene entry is identified by a unique

identifier within the "globalSceneEntryID" attribute.

```
<!-- GLOBAL SCENE ENTRY TYPE -->
<xs:complexType name="globalSceneEntryType">
  <xs:sequence>
    <xs:element name="sceneEntryIDREF" type="xs:IDREF"
      minOccurs="0" maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="globalSceneEntryID" type="xs:ID"/>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
```

19. <people>

Information about the participants that are represented in the media captures is conveyed via the <people> element. As it can be seen from the XML Schema depicted below, for each participant, a <person> element is provided.

```
<!-- PEOPLE TYPE -->
<xs:complexType name="peopleType">
  <xs:sequence>
    <xs:element name="person" type="personType"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- PERSON TYPE -->
<xs:complexType name="personType">
  <xs:sequence>
    <xs:element name="personInfo" type="xcard:vcardType" maxOccurs="1"
      minOccurs="0"/>
    <xs:element name="personType" type="personTypeType"
      minOccurs="0"
      maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="personID" type="xs:ID" use="required"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- PERSON TYPE TYPE -->
<xs:simpleType name="personTypeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="chairman"/>
    <xs:enumeration value="vice-chairman"/>
    <xs:enumeration value="minute taker"/>
    <xs:enumeration value="presenter"/>
    <xs:enumeration value="translator"/>
    <xs:enumeration value="timekeeper"/>
    <xs:enumeration value="attendee"/>
  </xs:restriction>
</xs:simpleType>
```

19.1. <person>

<person> includes all the metadata related to a person represented within one or more media captures. Such element provides the vcard of the subject (via the <personInfo> element, see Section 19.1.2) and his conference role(s) (via one or more <personType> elements, see Section 19.1.3). Furthermore, it has a mandatory "personID" attribute (Section 19.1.1).

19.1.1.1. personID attribute

The "personID" attribute carries the identifier of a represented person. Such identifier can be used to refer to the participant, as in the <capturedPeople> element in media captures representation (Section 10.19).

19.1.1.2. <personInfo>

The <personInfo> element is the XML representation of all the fields composing a vcard as specified in the Xcard RFC [RFC6351]. The vcardType is imported by the Xcard XML Schema provided by [I-D.ietf-ecrit-additional-data]. As such schema specifies, the <fn> element within <vcard> is mandatory.

19.1.1.3. <personType>

The value of the <personType> element determines the role of the represented participant within the telepresence session organization. It can be one of the following terms, that are defined in the framework document: "presenter", "timekeeper", "attendee", "minute taker", "translator", "chairman", "vice-chairman".

A participant can have more than one conference role. In that case, more than one <personType> element will appear in his description.

20. <captureEncoding>

A <captureEncoding> is given from the association of a media capture and an individual encoding, to form a capture stream as defined in [I-D.ietf-clue-framework]. The model of such an entity is provided in the following.

```
<!-- CAPTURE ENCODING TYPE -->
<xs:complexType name="captureEncodingType">
  <xs:sequence>
    <xs:element name="captureID" type="xs:string"/>
    <xs:element name="encodingID" type="xs:string"/>
    <xs:element name="configuredContent" type="contentType" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="ID" type="xs:ID" use="required"/>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
```

20.1. <captureID>

<captureID> is the mandatory element containing the identifier of the media capture that has been encoded to form the capture encoding.

20.2. <encodingID>

<encodingID> is the mandatory element containing the identifier of the applied individual encoding.

20.3. <configuredContent>

<configuredContent> is an optional element to be used in case of configuration of MCCs. It contains the list of capture identifiers and capture scene entry identifiers the Media Consumer wants within the MCC. That element is structured as the <content> element used to describe the content of a MCC, i.e. it contains The total number of the media captures listed in the <configuredContent> must be lower than or equal to the value carried within the <maxCaptures> attribute of the MCC.

21. <clueInfo>

The <clueInfo> element has been left within the XML Schema for representing a drafty version of the body of an ADVERTISEMENT message (see the example section).

```
<!-- CLUE INFO ELEMENT -->
<!-- the <clueInfo> envelope can be seen
      as the ancestor of an <advertisement> envelope -->
<xs:element name="clueInfo" type="clueInfoType"/>

<!-- CLUE INFO TYPE -->
<xs:complexType name="clueInfoType">
  <xs:sequence>
    <xs:element ref="mediaCaptures"/>
    <xs:element ref="encodingGroups"/>
    <xs:element ref="captureScenes"/>
    <xs:element ref="simultaneousSets" minOccurs="0"/>
    <xs:element ref="globalSceneEntries" minOccurs="0"/>
    <xs:element ref="people" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="clueInfoID" type="xs:ID" use="required"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

```
</xs:complexType>
```

22. XML Schema extensibility

The telepresence data model defined in this document is meant to be extensible. Extensions are accomplished by defining elements or attributes qualified by namespaces other than "urn:ietf:params:xml:ns:clue-info" and "urn:ietf:params:xml:ns:vcard-4.0" for use wherever the schema allows such extensions (i.e., where the XML Schema definition specifies "anyAttribute" or "anyElement"). Elements or attributes from unknown namespaces MUST be ignored.

23. Sample XML file

The following XML document represents a schema compliant example of a CLUE telepresence scenario. Taking inspiration from the examples described in the framework draft ([I-D.ietf-clue-framework]), it is provided the XML representation of an endpoint-style Media Provider's offer.

There are three cameras, where the central one is also able of capturing a zoomed-out view of the overall telepresence room. Besides the three video captures coming from such cameras, the MP makes available a further multi-content capture about the loudest segment of the room, obtained by switching the video source across the three cameras. For the sake of simplicity, only one audio capture is advertised for the audio of the whole room.

The three cameras are placed in front of three participants (Alice, Bob and Ciccio), whose vcard and conference roles details are also provided.

Media captures are arranged into four capture scene entries:

1. (VC0, VC1, VC2) - left, center and right camera video captures
2. (VC3) - video capture associated with loudest room segment
3. (VC4) - video capture zoomed out view of all people in the room
4. (AC0) - main audio

There are two encoding groups: (i) EG0, for video encodings, and (ii) EG1, for audio encodings.

As to the simultaneous sets, only VC1 and VC4 cannot be transmitted

simultaneously since they are captured by the same device, i.e., the central camera (VC4 is a zoomed-out view while VC1 is a focused view of the front participant). The simultaneous sets would then be the following:

SS1 made by VC3 and all the captures in the first capture scene entry (VC0,VC1,VC2);

SS2 made by VC3, VC0, VC2, VC4

Fri_Jun_13_16_16_16_CEST_2014_mccExample

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<clueInfo xmlns="urn:ietf:params:xml:ns:clue-info"
xmlns:ns2="urn:ietf:params:xml:ns:vcard-4.0" clueInfoID="NapoliRoom">
  <mediaCaptures>
    <mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="AC0">
      <capturedMedia>audio</capturedMedia>
      <captureSceneIDREF>CS1</captureSceneIDREF>
      <encGroupIDREF>EG1</encGroupIDREF>
      <spatialInformation>
        <capturePoint>
          <x>0.5</x>
          <y>1.0</y>
          <z>0.5</z>
          <lineOfCapturePoint>
            <x>0.5</x>
            <y>0.0</y>
            <z>0.5</z>
          </lineOfCapturePoint>
        </capturePoint>
      </spatialInformation>
      <individual>true</individual>
      <description lang="en">main audio from the room</description>
      <priority>1</priority>
      <lang>it</lang>
      <mobility>static</mobility>
      <view>room</view>
      <capturedPeople>
        <personIDREF>alice</personIDREF>
        <personIDREF>bob</personIDREF>
        <personIDREF>ciccio</personIDREF>
      </capturedPeople>
      <maxCaptureEncodings>1</maxCaptureEncodings>
    </mediaCapture>
  </mediaCaptures>
</clueInfo>
```

```
<mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="VC0">
  <capturedMedia>video</capturedMedia>
  <captureSceneIDREF>CS1</captureSceneIDREF>
  <encGroupIDREF>EG0</encGroupIDREF>
  <spatialInformation>
    <capturePoint>
      <x>0.5</x>
      <y>1.0</y>
      <z>0.5</z>
      <lineOfCapturePoint>
        <x>0.5</x>
        <y>0.0</y>
        <z>0.5</z>
      </lineOfCapturePoint>
    </capturePoint>
  </spatialInformation>
  <individual>true</individual>
  <description lang="en">left camera video capture</description>
  <priority>1</priority>
  <lang>it</lang>
  <mobility>static</mobility>
  <view>individual</view>
  <capturedPeople>
    <personIDREF>ciccio</personIDREF>
  </capturedPeople>
  <maxCaptureEncodings>2</maxCaptureEncodings>
</mediaCapture>
<mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="VC1">
  <capturedMedia>video</capturedMedia>
  <captureSceneIDREF>CS1</captureSceneIDREF>
  <encGroupIDREF>EG0</encGroupIDREF>
  <spatialInformation>
    <capturePoint>
      <x>0.5</x>
      <y>1.0</y>
      <z>0.5</z>
      <lineOfCapturePoint>
        <x>0.5</x>
        <y>0.0</y>
        <z>0.5</z>
      </lineOfCapturePoint>
    </capturePoint>
  </spatialInformation>
  <individual>true</individual>
  <description lang="en">central camera video capture</description>
  <priority>1</priority>
```

```
<lang>it</lang>
<mobility>static</mobility>
<view>individual</view>
<capturedPeople>
  <personIDREF>alice</personIDREF>
</capturedPeople>
<maxCaptureEncodings>2</maxCaptureEncodings>
</mediaCapture>
<mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="VC2">
  <capturedMedia>video</capturedMedia>
  <captureSceneIDREF>CS1</captureSceneIDREF>
  <encGroupIDREF>EG0</encGroupIDREF>
  <spatialInformation>
    <capturePoint>
      <x>0.5</x>
      <y>1.0</y>
      <z>0.5</z>
      <lineOfCapturePoint>
        <x>0.5</x>
        <y>0.0</y>
        <z>0.5</z>
      </lineOfCapturePoint>
    </capturePoint>
  </spatialInformation>
  <individual>true</individual>
  <description lang="en">right camera video capture</description>
  <priority>1</priority>
  <lang>it</lang>
  <mobility>static</mobility>
  <view>individual</view>
  <capturedPeople>
    <personIDREF>bob</personIDREF>
  </capturedPeople>
  <maxCaptureEncodings>2</maxCaptureEncodings>
</mediaCapture>
<mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="VC3">
  <capturedMedia>video</capturedMedia>
  <captureSceneIDREF>CS1</captureSceneIDREF>
  <encGroupIDREF>EG0</encGroupIDREF>
  <nonSpatiallyDefinable>true</nonSpatiallyDefinable>
  <composed>false</composed>
  <switched>true</switched>
  <policy>Soundlevel:0</policy>
  <maxCaptures>1</maxCaptures>
  <description lang="en">loudest room segment</description>
  <priority>1</priority>
```

```
<lang>it</lang>
<mobility>static</mobility>
<view>individual</view>
<maxCaptureEncodings>1</maxCaptureEncodings>
</mediaCapture>
<mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="VC4">
  <capturedMedia>video</capturedMedia>
  <captureSceneIDREF>CS1</captureSceneIDREF>
  <encGroupIDREF>EG0</encGroupIDREF>
  <spatialInformation>
    <capturePoint>
      <x>0.5</x>
      <y>1.0</y>
      <z>0.5</z>
      <lineOfCapturePoint>
        <x>0.5</x>
        <y>0.0</y>
        <z>0.5</z>
      </lineOfCapturePoint>
    </capturePoint>
  </spatialInformation>
  <individual>true</individual>
  <description lang="en">zoomed out view of all people in the room
</description>
  <priority>1</priority>
  <lang>it</lang>
  <mobility>static</mobility>
  <view>room</view>
  <capturedPeople>
    <personIDREF>alice</personIDREF>
    <personIDREF>bob</personIDREF>
    <personIDREF>ciccio</personIDREF>
  </capturedPeople>
  <maxCaptureEncodings>1</maxCaptureEncodings>
</mediaCapture>
</mediaCaptures>
<encodingGroups>
  <encodingGroup encodingGroupID="EG0">
    <maxGroupBandwidth>600000</maxGroupBandwidth>
    <encodingIDList>
      <encID>ENC1</encID>
      <encID>ENC2</encID>
      <encID>ENC3</encID>
    </encodingIDList>
  </encodingGroup>
  <encodingGroup encodingGroupID="EG1">
    <maxGroupBandwidth>300000</maxGroupBandwidth>
```

```
<encodingIDList>
  <encID>ENC4</encID>
  <encID>ENC5</encID>
</encodingIDList>
</encodingGroup>
</encodingGroups>
<captureScenes>
  <captureScene scale="unknown" sceneID="CS1">
    <sceneEntries>
      <sceneEntry mediaType="video" sceneEntryID="SE1">
        <mediaCaptureIDs>
          <captureIDREF>VC0</captureIDREF>
          <captureIDREF>VC1</captureIDREF>
          <captureIDREF>VC2</captureIDREF>
        </mediaCaptureIDs>
      </sceneEntry>
      <sceneEntry mediaType="video" sceneEntryID="SE2">
        <mediaCaptureIDs>
          <captureIDREF>VC3</captureIDREF>
        </mediaCaptureIDs>
      </sceneEntry>
      <sceneEntry mediaType="video" sceneEntryID="SE3">
        <mediaCaptureIDs>
          <captureIDREF>VC4</captureIDREF>
        </mediaCaptureIDs>
      </sceneEntry>
      <sceneEntry mediaType="audio" sceneEntryID="SE4">
        <mediaCaptureIDs>
          <captureIDREF>AC0</captureIDREF>
        </mediaCaptureIDs>
      </sceneEntry>
    </sceneEntries>
  </captureScene>
</captureScenes>
<simultaneousSets>
  <simultaneousSet setID="SS1">
    <captureIDREF>VC3</captureIDREF>
    <sceneEntryIDREF>SE1</sceneEntryIDREF>
  </simultaneousSet>
  <simultaneousSet setID="SS2">
    <captureIDREF>VC0</captureIDREF>
    <captureIDREF>VC2</captureIDREF>
    <captureIDREF>VC4</captureIDREF>
    <captureIDREF>VC3</captureIDREF>
  </simultaneousSet>
</simultaneousSets>
<people>
  <person personID="bob">
```

```
        <personInfo>
          <ns2:fn>
            <ns2:text>Bob</ns2:text>
          </ns2:fn>
        </personInfo>
        <personType>minute taker</personType>
      </person>
    <person personID="alice">
      <personInfo>
        <ns2:fn>
          <ns2:text>Alice</ns2:text>
        </ns2:fn>
      </personInfo>
      <personType>presenter</personType>
    </person>
    <person personID="ciccio">
      <personInfo>
        <ns2:fn>
          <ns2:text>Ciccio</ns2:text>
        </ns2:fn>
      </personInfo>
      <personType>chairman</personType>
      <personType>timekeeper</personType>
    </person>
  </people>
</clueInfo>
```

24. MCC example

Enhancing the scenario presented in the previous example, the Media Providers is able to advertise a composed capture VC7 made by a big picture representing the current speaker (VC3) and two picture-in-picture boxes representing the previous speakers (the previous one -VC5- and the oldest one -VC6). The provider does not want to instantiate and send VC5 and VC6, so it does not associate any encoding group with them. Their XML representations are provided for enabling the description of VC7.

A possible description for that scenario could be the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<clueInfo xmlns="urn:ietf:params:xml:ns:clue-info"
  xmlns:ns2="urn:ietf:params:xml:ns:vcard-4.0"
  clueInfoID="NapoliRoom">
```

```
<mediaCaptures>
  <mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="AC0">
    <capturedMedia>audio</capturedMedia>
    <captureSceneIDREF>CS1</captureSceneIDREF>
    <encGroupIDREF>EG1</encGroupIDREF>
    <spatialInformation>
      <capturePoint>
        <x>0.5</x>
        <y>1.0</y>
        <z>0.5</z>
        <lineOfCapturePoint>
          <x>0.5</x>
          <y>0.0</y>
          <z>0.5</z>
        </lineOfCapturePoint>
      </capturePoint>
    </spatialInformation>
    <individual>true</individual>
    <description lang="en">main audio from the room</description>
    <priority>1</priority>
    <lang>it</lang>
    <mobility>static</mobility>
    <view>room</view>
    <capturedPeople>
      <personIDREF>alice</personIDREF>
      <personIDREF>bob</personIDREF>
      <personIDREF>ciccio</personIDREF>
    </capturedPeople>
    <maxCaptureEncodings>1</maxCaptureEncodings>
  </mediaCapture>
  <mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="VC0">
    <capturedMedia>video</capturedMedia>
    <captureSceneIDREF>CS1</captureSceneIDREF>
    <encGroupIDREF>EG0</encGroupIDREF>
    <spatialInformation>
      <capturePoint>
        <x>0.5</x>
        <y>1.0</y>
        <z>0.5</z>
        <lineOfCapturePoint>
          <x>0.5</x>
          <y>0.0</y>
          <z>0.5</z>
        </lineOfCapturePoint>
      </capturePoint>
    </spatialInformation>
```

```
<individual>true</individual>
<description lang="en">left camera video capture</description>
<priority>1</priority>
<lang>it</lang>
<mobility>static</mobility>
<view>individual</view>
<capturedPeople>
  <personIDREF>ciccio</personIDREF>
</capturedPeople>
<maxCaptureEncodings>2</maxCaptureEncodings>
</mediaCapture>
<mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="VC1">
  <capturedMedia>video</capturedMedia>
  <captureSceneIDREF>CS1</captureSceneIDREF>
  <encGroupIDREF>EG0</encGroupIDREF>
  <spatialInformation>
    <capturePoint>
      <x>0.5</x>
      <y>1.0</y>
      <z>0.5</z>
      <lineOfCapturePoint>
        <x>0.5</x>
        <y>0.0</y>
        <z>0.5</z>
      </lineOfCapturePoint>
    </capturePoint>
  </spatialInformation>
  <individual>true</individual>
  <description lang="en">central camera video capture</description>
  <priority>1</priority>
  <lang>it</lang>
  <mobility>static</mobility>
  <view>individual</view>
  <capturedPeople>
    <personIDREF>alice</personIDREF>
  </capturedPeople>
  <maxCaptureEncodings>2</maxCaptureEncodings>
</mediaCapture>
<mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="VC2">
  <capturedMedia>video</capturedMedia>
  <captureSceneIDREF>CS1</captureSceneIDREF>
  <encGroupIDREF>EG0</encGroupIDREF>
  <spatialInformation>
    <capturePoint>
      <x>0.5</x>
      <y>1.0</y>
```



```
        <z>0.5</z>
        <lineOfCapturePoint>
            <x>0.5</x>
            <y>0.0</y>
            <z>0.5</z>
        </lineOfCapturePoint>
    </capturePoint>
</spatialInformation>
<individual>true</individual>
<description lang="en">right camera video capture</description>
<priority>1</priority>
<lang>it</lang>
<mobility>static</mobility>
<view>individual</view>
<capturedPeople>
    <personIDREF>bob</personIDREF>
</capturedPeople>
<maxCaptureEncodings>2</maxCaptureEncodings>
</mediaCapture>
<mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="VC3">
    <capturedMedia>video</capturedMedia>
    <captureSceneIDREF>CS1</captureSceneIDREF>
    <encGroupIDREF>EG0</encGroupIDREF>
    <nonSpatiallyDefinable>true</nonSpatiallyDefinable>
    <content>
        <sceneEntryIDREF>SE1</sceneEntryIDREF>
    </content>
    <composed>false</composed>
    <switched>true</switched>
    <policy>Soundlevel:0</policy>
    <maxCaptures>1</maxCaptures>
    <description lang="en">loudest room segment</description>
    <priority>1</priority>
    <lang>it</lang>
    <mobility>static</mobility>
    <view>individual</view>
    <maxCaptureEncodings>1</maxCaptureEncodings>
</mediaCapture>
<mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="VC4">
    <capturedMedia>video</capturedMedia>
    <captureSceneIDREF>CS1</captureSceneIDREF>
    <encGroupIDREF>EG0</encGroupIDREF>
    <spatialInformation>
        <capturePoint>
            <x>0.5</x>
            <y>1.0</y>
```

```
<z>0.5</z>
<lineOfCapturePoint>
  <x>0.5</x>
  <y>0.0</y>
  <z>0.5</z>
</lineOfCapturePoint>
</capturePoint>
</spatialInformation>
<individual>true</individual>
<description lang="en">zoomed out view of all people in the room
</description>
<priority>1</priority>
<lang>it</lang>
<mobility>static</mobility>
<view>room</view>
<capturedPeople>
  <personIDREF>alice</personIDREF>
  <personIDREF>bob</personIDREF>
  <personIDREF>ciccio</personIDREF>
</capturedPeople>
<maxCaptureEncodings>1</maxCaptureEncodings>
</mediaCapture>
<mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="VC5">
  <capturedMedia>video</capturedMedia>
  <captureSceneIDREF>CS1</captureSceneIDREF>
  <nonSpatiallyDefinable>true</nonSpatiallyDefinable>
  <content>
    <sceneEntryIDREF>SE1</sceneEntryIDREF>
  </content>
  <composed>false</composed>
  <switched>true</switched>
  <policy>Soundlevel:1</policy>
  <maxCaptures>1</maxCaptures>
  <description lang="en">penultimate loudest room segment
  </description>
  <priority>1</priority>
  <lang>it</lang>
  <mobility>static</mobility>
  <view>individual</view>
  <maxCaptureEncodings>1</maxCaptureEncodings>
</mediaCapture>
<mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="VC6">
  <capturedMedia>video</capturedMedia>
  <captureSceneIDREF>CS1</captureSceneIDREF>
  <nonSpatiallyDefinable>true</nonSpatiallyDefinable>
  <content>
```

```
        <sceneEntryIDREF>SE1</sceneEntryIDREF>
    </content>
    <composed>false</composed>
    <switched>true</switched>
    <policy>Soundlevel:2</policy>
    <maxCaptures>1</maxCaptures>
    <description lang="en">last but two loudest room segment
    </description>
    <priority>1</priority>
    <lang>it</lang>
    <mobility>static</mobility>
    <view>individual</view>
    <maxCaptureEncodings>1</maxCaptureEncodings>
</mediaCapture>
<mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="videoCaptureType" captureID="VC7">
    <capturedMedia>video</capturedMedia>
    <captureSceneIDREF>CS1</captureSceneIDREF>
    <nonSpatiallyDefinable>true</nonSpatiallyDefinable>
    <content>
        <captureIDREF>VC3</captureIDREF>
        <captureIDREF>VC5</captureIDREF>
        <captureIDREF>VC6</captureIDREF>
    </content>
    <composed>true</composed>
    <switched>true</switched>
    <maxCaptures>1</maxCaptures>
    <description lang="en">big picture of the current speaker + pips
    about previous speakers</description>
    <priority>1</priority>
    <lang>it</lang>
    <mobility>static</mobility>
    <view>individual</view>
    <maxCaptureEncodings>1</maxCaptureEncodings>
</mediaCapture>
</mediaCaptures>
<encodingGroups>
    <encodingGroup encodingGroupID="EG0">
        <maxGroupBandwidth>600000</maxGroupBandwidth>
        <encodingIDList>
            <encID>ENC1</encID>
            <encID>ENC2</encID>
            <encID>ENC3</encID>
        </encodingIDList>
    </encodingGroup>
    <encodingGroup encodingGroupID="EG1">
        <maxGroupBandwidth>300000</maxGroupBandwidth>
        <encodingIDList>
```

```
<encID>ENC4</encID>
<encID>ENC5</encID>
</encodingIDList>
</encodingGroup>
</encodingGroups>
<captureScenes>
  <captureScene scale="unknown" sceneID="CS1">
    <sceneEntries>
      <sceneEntry mediaType="video" sceneEntryID="SE1">
        <description lang="en">participants' individual videos
        </description>
        <mediaCaptureIDs>
          <captureIDREF>VC0</captureIDREF>
          <captureIDREF>VC1</captureIDREF>
          <captureIDREF>VC2</captureIDREF>
        </mediaCaptureIDs>
      </sceneEntry>
      <sceneEntry mediaType="video" sceneEntryID="SE2">
        <description lang="en">loudest segment of the room
        </description>
        <mediaCaptureIDs>
          <captureIDREF>VC3</captureIDREF>
        </mediaCaptureIDs>
      </sceneEntry>
      <sceneEntry mediaType="video" sceneEntryID="SE5">
        <description lang="en">loudest segment of the room + pips
        </description>
        <mediaCaptureIDs>
          <captureIDREF>VC7</captureIDREF>
        </mediaCaptureIDs>
      </sceneEntry>
      <sceneEntry mediaType="audio" sceneEntryID="SE4">
        <description lang="en">room audio</description>
        <mediaCaptureIDs>
          <captureIDREF>AC0</captureIDREF>
        </mediaCaptureIDs>
      </sceneEntry>
      <sceneEntry mediaType="video" sceneEntryID="SE3">
        <description lang="en">room video</description>
        <mediaCaptureIDs>
          <captureIDREF>VC4</captureIDREF>
        </mediaCaptureIDs>
      </sceneEntry>
    </sceneEntries>
  </captureScene>
</captureScenes>
<simultaneousSets>
  <simultaneousSet setID="SS1">
```

```
        <captureIDREF>VC7</captureIDREF>
        <sceneEntryIDREF>SE1</sceneEntryIDREF>
    </simultaneousSet>
    <simultaneousSet setID="SS2">
        <captureIDREF>VC0</captureIDREF>
        <captureIDREF>VC2</captureIDREF>
        <captureIDREF>VC4</captureIDREF>
        <captureIDREF>VC7</captureIDREF>
    </simultaneousSet>
</simultaneousSets>
<people>
    <person personID="bob">
        <personInfo>
            <ns2:fn>
                <ns2:text>Bob</ns2:text>
            </ns2:fn>
        </personInfo>
        <personType>minute taker</personType>
    </person>
    <person personID="alice">
        <personInfo>
            <ns2:fn>
                <ns2:text>Alice</ns2:text>
            </ns2:fn>
        </personInfo>
        <personType>presenter</personType>
    </person>
    <person personID="ciccio">
        <personInfo>
            <ns2:fn>
                <ns2:text>Ciccio</ns2:text>
            </ns2:fn>
        </personInfo>
        <personType>chairman</personType>
        <personType>timekeeper</personType>
    </person>
</people>
</clueInfo>
```

25. Diff with draft-ietf-clue-data-model-schema-02 version

captureParameters and encodingParameters have been removed from the captureEncodingType

data model example has been updated and validated according to the new schema. Further description of the represented scenario have been provided.

A multiple content capture example has been added.

Obsolete comments and references have been removed.

26. Diff with draft-ietf-clue-data-model-schema-03 version

encodings section has been removed

global capture entries have been introduced

capture scene entry identifiers are used as shortcuts in listing the content of MCC (similarly to simultaneous set and global capture entries)

Examples have been updated. A new example with global capture entries has been added.

<encGroupIDREF> has been made optional.

<single> has been renamed into <individual>

Obsolete comments have been removed.

participants information has been added.

27. Diff with draft-ietf-clue-data-model-schema-04 version

globalCaptureEntries/Entry renamed as globalSceneEntries/Entry;

sceneInformation added;

Only capture scene entry identifiers listed within global scene entries (media capture identifiers removed);

<participants> renamed as <people> in the >clueInfo< template

<vcard> renamed as <personInfo> to synch with the framework terminology

<participantType> renamed as <personType> to synch with the framework terminology

<participantIDs> renamed as <capturedPeople> in the media capture type definition to remove ambiguity

Examples have been updated with the new definitions of <globalSceneEntries> and of <people>.

28. Informative References

- [I-D.ietf-clue-framework] Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-15 (work in progress), May 2014.
- [I-D.ietf-clue-protocol] Presta, R. and S. Romano, "CLUE protocol", draft-ietf-clue-protocol-01 (work in progress), June 2014.
- [I-D.ietf-ecrit-additional-data] Rosen, B., Tschofenig, H., Marshall, R., Randy, R., and J. Winterbottom, "Additional Data related to an Emergency Call", draft-ietf-ecrit-additional-data-22 (work in progress), April 2014.
- [RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, February 2007.
- [RFC6351] Perreault, S., "xCard: vCard XML Representation", RFC 6351, August 2011.

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 Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 21, 2014

C. Holmberg
Ericsson
March 20, 2014

CLUE Protocol Data Channel
draft-ietf-clue-datachannel-00

Abstract

This document defines how to use the WebRTC Data Channel mechanism, together with the Data Channel Establishment Protocol (DCEP) in order to establish a data channel, referred to as CLUE Data Channel, for transporting CLUE protocol messages between two CLUE entities.

The document defines the SCTP considerations specific to a CLUE Data Channel, the SDP offer/answer procedures for negotiating the establishment of, and the DCEP procedures for opening, a CLUE Data Channel.

Details and procedures associated with the CLUE protocol are outside the scope of this document.

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 September 21, 2014.

Copyright Notice

Copyright (c) 2014 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. Conventions	3
3. CLUE Data Channel	4
3.1. General	4
3.2. Data Channel Establishment Protocol (DCEP) Usage	4
3.3. SCTP Considerations	4
3.3.1. SCTP Payload Protocol Identifier (PPID)	4
3.3.2. Reliability	5
3.3.3. Order	5
3.3.4. Stream Reset	5
3.3.5. Interleaving	5
3.3.6. SCTP Multihoming	6
4. CLUE Data Channel Procedures	6
4.1. Open CLUE Data Channel	6
4.2. Close CLUE Data Channel	6
4.3. SCTP Association Failure	7
5. SDP Offer/Answer Procedures	7
5.1. General	7
5.2. SDP Media Description Fields	7
5.3. SDP sctpmap Attribute	8
5.4. SDP Offerer Procedures	8
5.5. SDP Answerer Procedures	8
5.6. Example	9
6. Security Considerations	9
7. IANA Considerations	9
8. Acknowledgments	9
9. Change Log	9
10. References	10
10.1. Normative References	10
10.2. Informative References	12
Author's Address	12

1. Introduction

This document defines how to use the WebRTC Data Channel mechanism [I-D.ietf-rtcweb-data-channel], together with the Data Channel Establishment Protocol (DCEP) [I-D.ietf-rtcweb-data-protocol] in order to establish a data channel, referred to as CLUE Data Channel,

for transporting CLUE protocol [I-D.presta-clue-protocol] messages between CLUE entities.

The document defines the SCTP considerations specific to a CLUE Data Channel, the SDP offer/answer [RFC3264] procedures for negotiating the establishment of, and the DCEP procedures for opening, a CLUE Data Channel.

Details and procedures associated with the CLUE protocol are outside the scope of this document.

2. Conventions

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 BCP 14, RFC 2119 [RFC2119].

WebRTC Data Channel refers to a SCTPoDTLS association [I-D.ietf-tsvwg-sctp-dtls-encaps] that is used to transport non-media data between two entities, according to the procedures in [I-D.ietf-rtcweb-data-channel].

CLUE Data Channel refers to a WebRTC Data Channel [I-D.ietf-rtcweb-data-channel], with a specific set of SCTP characteristics, and usage of the Data Channel Establishment Protocol (DCEP) [I-D.ietf-rtcweb-data-protocol] in order to open a WebRTC Data Channel for the purpose of transporting CLUE protocol [I-D.presta-clue-protocol] messages between two CLUE entities.

CLUE entity refers to a SIP User Agent (UA) [RFC3261] that supports the CLUE Data Channel and the CLUE protocol.

CLUE session refers to a SIP session [RFC3261] between to SIP UAs, where a CLUE Data Channel, associated with the SIP session, has been established between the SIP UAs.

[RFC4960] defines an SCTP stream as a unidirectional logical channel established from one to another associated SCTP endpoint, within which all user messages are delivered in sequence except for those submitted to the unordered delivery service.

[RFC4960] defines an SCTP identifier as a unsigned integer, which identifies a SCTP stream.

3. CLUE Data Channel

3.1. General

This section describes the realization of a CLUE Data Channel. This includes a set of SCTP characteristics specific to a CLUE Data Channel, and usage of the Data Channel Establishment Protocol (DCEP) [I-D.ietf-rtcweb-data-protocol] in order to open a WebRTC Data Channel for the purpose of transporting CLUE protocol [I-D.presta-clue-protocol] messages between two CLUE entities.

As described in [I-D.ietf-rtcweb-data-channel], the SCTP streams realizing a WebRTC Data Channel must be associated with the same SCTP association. In addition, both SCTP streams realizing the WebRTC Data Channel must use the same SCTP stream identifier value. These rules also apply to a CLUE Data Channel.

Within a given CLUE session, a CLUE entity **MUST** use a single CLUE Data Channel for transport of all CLUE messages towards its peer.

3.2. Data Channel Establishment Protocol (DCEP) Usage

A CLUE entity **MUST** support the Data Channel Establishment Protocol (DCEP) [I-D.ietf-rtcweb-data-channel], which can be used in order to open a WebRTC Data Channel.

In the absence of some other mechanism, a CLUE entity **MUST** use DCEP in order to open a CLUE Data Channel.

NOTE: This document does not define any other mechanism for opening a CLUE Data Channel, but such might be defined in future specifications.

The details of the DCEP usage with a CLUE Data Channel are described in Section 4.1.

3.3. SCTP Considerations

3.3.1. SCTP Payload Protocol Identifier (PPID)

As described in [I-D.ietf-rtcweb-data-protocol], the PPID value 50 is used when sending a DCEP message on a WebRTC Data Channel.

A CLUE entity **MUST** use the PPID value 51 when sending a CLUE message on a CLUE Data Channel.

NOTE: As described in [I-D.ietf-rtcweb-data-channel], the PPID value 51 indicates that the SCTP message contains data encoded in a UTF-8

format. The PPID value 51 does not indicate what application protocol is transported in a WebRTC Data Channel, only the format in which the data is encoded.

Protocol	PPID Value
DCEP	50
CLUE	51

Table 1: CLUE Data Channel PPID Values

3.3.2. Reliability

The usage of SCTP for the CLUE Data Channel ensures reliable transport of CLUE protocol [I-D.presta-clue-protocol] messages.

A CLUE entity MUST NOT use the partial reliability and limited retransmission extensions defined in [RFC3758].

NOTE: [I-D.ietf-rtcweb-data-channel] requires the support of the partial reliability extension defined in [RFC3758]. This is not needed for a CLUE Data Channel, as messages are required to always be sent reliably. [I-D.ietf-rtcweb-data-channel] also mandates support of the limited retransmission policy defined in [I-D.tuexen-tsvwg-sctp-prpolicies].

3.3.3. Order

A CLUE entity MUST use the ordered delivery SCTP service, as described in section 6.6 of [RFC4960].

3.3.4. Stream Reset

A CLUE entity MUST support the stream reset extension defined in [RFC6525].

The dynamic address reconfiguration extension defined in [RFC5061] MUST be used to signal the support of the stream reset extension defined in [RFC6525]. Other features of [RFC5061] MUST NOT be used.

3.3.5. Interleaving

A CLUE entity MUST support the message interleaving mechanism defined in [I-D.stewart-tsvwg-sctp-ndata].

3.3.6. SCTP Multihoming

SCTP multihoming cannot be used for a CLUE Data Channel.

NOTE: SCTPoDTLS does not support SCTP multihoming.

4. CLUE Data Channel Procedures

4.1. Open CLUE Data Channel

Once the SCTP association, to be used to realized the CLUE Data Channel, has been established, the offerer [RFC3264] is responsible for opening the CLUE Data Channel. If DCEP is used, the offerer MUST send a DCEP DATA_CHANNEL_OPEN message [I-D.ietf-rtcweb-data-protocol]. The value of the 'protocol' field MUST be "CLUE". The value of the 'channel type' MUST be 'DATA_CHANNEL_RELIABLE'.

OPEN ISSUE: We need to determine whether we shall include a version number in the 'protocol' field value for CLUE.

NOTE: A new 'protocol' value for CLUE needs to be registered with IANA in the 'Protocol Registry' defined by [I-D.ietf-rtcweb-data-protocol].

Once the offerer has received the associated DCEP DATA_CHANNEL_ACK message [I-D.ietf-rtcweb-data-protocol], the CLUE Data channel has been opened.

If the offerer receives a DCEP DATA_CHANNEL_OPEN message, for the purpose of opening a CLUE Data Channel, the offerer MUST reset the SCTP stream, in order to prevent two CLUE Data Channels from being established within the same CLUE session. The offerer MUST NOT send a DCEP DATA_CHANNEL_ACK message.

4.2. Close CLUE Data Channel

DCEP [I-D.ietf-rtcweb-data-protocol] does not define a message for closing a WebRTC Data Channel. As described in [I-D.ietf-rtcweb-data-protocol], in order to close a CLUE Data Channel, a SCTP reset message is sent, in order to close the SCTP stream associated with the CLUE Data Channel. The SCTP association, and WebRTC Data Channels associated with other SCTP streams, are not affected by the SCTP reset message.

Section 5.4 describes how to terminate the SCTP association used for the CLUE data channel.

4.3. SCTP Association Failure

In case of SCTP association failure, the offerer is responsible for trying to re-establish the SCTP association (including sending a new SDP offer, if needed). Once the SCTP association has been successfully re-established, the offerer is responsible for sending a DCEP DATA_CHANNEL_OPEN message.

5. SDP Offer/Answer Procedures

5.1. General

This section describes how an SDP media description ("m=") line describing a SCTPoDTLS association, to be used to realize a CLUE Data Channel, is created, and how it is used in SDP offers and answers [RFC3264].

NOTE: The procedures associated with creating an "m=" line describing media (e.g. audio and video) for a CLUE session are outside the scope of this document.

OPEN ISSUE (Q1): It is FFS whether the SDP-based WebRTC Data Channel Negotiation mechanism [I-D.ejzak-dispatch-webrtc-data-channel-sdpneg] will be used with the CLUE Data Channel. It depends on whether the draft will progress in MMUSIC, and whether it will be finalized before the publication of the CLUE mechanism.

OPEN ISSUE (Q2): As the SDP offer/answer procedures are generic to SCTPoDTLS association, it is FFS whether we need to specify them, or whether we can simply refer to draft-ietf-mmusic-sctp-sdp.

5.2. SDP Media Description Fields

The field values of the "m=" line for the SCTPoDTLS association are set as following:

media	port	proto	fmt
"application"	DTLS port value	"DTLS/SCTP"	SCTP port value

Table 2: SDP "proto" field values

5.3. SDP sctpmap Attribute

The field values of the SDP sctpmap attribute, associated with the "m=" line describing the SCTPoDTLS association, are set as following:

sctpmap-number	app
fmt value of the "m=" line	"webrtc-datachannel"

Table 3: SDP "proto" field values

5.4. SDP Offerer Procedures

The procedures for the offerer follow the normal procedures defined in [RFC3264].

When the offerer creates an offer, which contains an "m=" line describing a SCTPoDTLS association, it assigns the field values to the "m=" line according to the procedures in Section 5.2. In addition, the offerer MUST insert an SDP sctpmap attribute associated with the "m=" line.

If an offerer, in a subsequent offer, wants to disable the CLUE Data Channel, it assigns a zero port value to the "m=" line describing the SCTPoDTLS association used to realize the CLUE Data Channel.

5.5. SDP Answerer Procedures

The procedures for the answerer follow the normal procedures defined in [RFC3264].

If the answerer receives an offer, which contains an "m=" line describing a SCTPoDTLS association, and the answerer accepts the "m=" line, it inserts an "m=" line in the corresponding answer, and assigns the "m=" line field values according to the procedures in Section 4.2.

If the answerer receives an offer, which contains an "m=" line describing a SCTPoDTLS association, and the answerer does not accept the "m=" line, it inserts an "m=" line in the corresponding answer, and assigns a zero port value to the "m=" line, according to the procedures in [RFC3264].

If the answerer receives an offer, in which a zero port value has been assigned to an "m=" line describing the SCTPoDTLS association, it inserts an "m=" line in the corresponding answer, and assigns a

zero port value to the "m=" line, according to the procedures in [RFC3264]

OPEN ISSUE (Q3): We need to determine whether an "m=" line describing an SCTPoDTLS association can be used together with bundle-only, in which case there will be cases where an offer with a zero port value will create a corresponding answer with a non-zero port value.

5.6. Example

```
m=application 54111 SCTP/DTLS 54111
a=sctpmap:54111 webrtc-datachannel
```

Figure 1: SDP Media Description for a CLUE Data Channel

6. Security Considerations

This specification does not introduce new security considerations, in addition to those defined in [ref-to-data-channel] and [ref-to-data-protocol]. Security considerations associated with the CLUE protocol are defined in [ref-to-clue-protocol].

7. IANA Considerations

[RFC EDITOR NOTE: Please replace RFC-XXXX with the RFC number of this document.]

8. Acknowledgments

Thanks to Paul Kyzivat and Christian Groves for comments on the document.

9. Change Log

[RFC EDITOR NOTE: Please remove this section when publishing]

Changes from draft-holmberg-clue-datachannel-04

- o Draft submitted as draft-ietf-clue-data-channel-00.
- o Editorial nits fixed.
- o Changes based on comments from Paul Kyzivat (<http://www.ietf.org/mail-archive/web/clue/current/msg03559.html>).
- o - Proto value fixed.
- o - Explicit text that the partial reliability and limited retransmission policies MUST NOT be used.
- o - Added open issue on whether the DCEP 'protocol' field value for CLUE should contain a version number.

- o - Removed paragraph saying that an offerer must not insert more than one m- line describing an SCTPoDTLS association to be used to realize a CLUE Data Channel, as the draft already states that only one CLUE Data Channel per CLUE session shall be opened.
- o - Added reference to draft-ietf-rtcweb-data-protocol regarding details on resetting SCTP streams.
- o - Added text saying that the value of the DCEP 'channel type' MUST be DATA_CHANNEL_RELIABLE.
- o - Clarified that DCEP must be supported, and used in the absence of another mechanism for opening a CLUE Data Channel.

Changes from draft-holmberg-clue-datachannel-03

- o Procedures updated, based on WG agreement (IETF#89) to use DCEP for the CLUE data channel.
- o Procedures updated, based on WG agreement (IETF#89) that offerer is responsible for sending DCEP DATA_CHANNEL_OPEN.
- o Editorial changes, and alignments caused by changes in referenced specifications.

Changes from draft-holmberg-clue-datachannel-02

- o PPID value for CLUE messages added
- o References updated

Changes from draft-holmberg-clue-datachannel-01

- o More text added

Changes from draft-holmberg-clue-datachannel-00

- o Editorial corrections based on comments from Paul K

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.
- [RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, February 2012.
- [I-D.presta-clue-protocol]
Presta, R. and S. Romano, "CLUE protocol", draft-presta-clue-protocol-03.txt (work in progress), November 2013.
- [I-D.ietf-tsvwg-sctp-dtls-encaps]
Tuexen, M., Stewart, R., Jesup, R., and S. Loreto, "DTLS Encapsulation of SCTP Packets", draft-ietf-tsvwg-sctp-dtls-encaps-03.txt (work in progress), February 2014.
- [I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-07.txt (work in progress), February 2014.
- [I-D.ietf-rtcweb-data-protocol]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel Establishment Protocol", draft-ietf-rtcweb-data-protocol-03.txt (work in progress), February 2014.
- [I-D.stewart-tsvwg-sctp-ndata]
Stewart, R., Tuexen, M., Loreto, S., and R. Seggelmann, "A New Data Chunk for Stream Control Transmission Protocol", draft-stewart-tsvwg-sctp-ndata-03.txt (work in progress), October 2013.
- [I-D.tuexen-tsvwg-sctp-prpolicies]
Tuexen, M., Seggelmann, R., Stewart, R., and S. Loreto, "Additional Policies for the Partial Delivery Extension of the Stream Control Transmission Protocol", draft-tuexen-tsvwg-sctp-prpolicies-03.txt (work in progress), October 2013.

10.2. Informative References

- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.
- [I-D.ejzak-dispatch-webrtc-data-channel-sdpneg]
Ejzak, R. and J. Marcon, "SDP-based WebRTC data channel negotiation", draft-ejzak-dispatch-webrtc-data-channel-sdpneg-00.txt (work in progress), October 2013.

Author's Address

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com

CLUE WG
Internet Draft
Intended status: Standards Track
Expires: December 27, 2014

M. Duckworth, Ed.
Polycom
A. Pepperell
Acano
S. Wenger
Vidyo
June 27, 2014

Framework for Telepresence Multi-Streams
draft-ietf-clue-framework-16.txt

Abstract

This document defines a framework for a protocol to enable devices in a telepresence conference to interoperate. The protocol enables communication of information about multiple media streams so a sending system and receiving system can make reasonable decisions about transmitting, selecting and rendering the media streams. This protocol is used in addition to SIP signaling for setting up a telepresence session.

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 November 15, 2014.

Copyright Notice

Copyright (c) 2013 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.....	4
3. Definitions.....	4
4. Overview & Motivation.....	7
5. Overview of the Framework/Model.....	9
6. Spatial Relationships.....	14
7. Media Captures and Capture Scenes.....	16
7.1. Media Captures.....	16
7.1.1. Media Capture Attributes.....	17
7.2. Multiple Content Capture.....	23
7.2.1. MCC Attributes.....	24
7.3. Capture Scene.....	29
7.3.1. Capture Scene attributes.....	31
7.3.2. Capture Scene Entry attributes.....	32
7.3.3. Global Capture Scene Entry List.....	32
8. Simultaneous Transmission Set Constraints.....	33
9. Encodings.....	35
9.1. Individual Encodings.....	35
9.2. Encoding Group.....	36
9.3. Associating Captures with Encoding Groups.....	37
10. Consumer's Choice of Streams to Receive from the Provider....	38
10.1. Local preference.....	41
10.2. Physical simultaneity restrictions.....	41
10.3. Encoding and encoding group limits.....	41
11. Extensibility.....	42
12. Examples - Using the Framework (Informative).....	42
12.1. Provider Behavior.....	42
12.1.1. Three screen Endpoint Provider.....	42
12.1.2. Encoding Group Example.....	49
12.1.3. The MCU Case.....	50

12.2. Media Consumer Behavior.....	51
12.2.1. One screen Media Consumer.....	51
12.2.2. Two screen Media Consumer configuring the example..	52
12.2.3. Three screen Media Consumer configuring the example	53
12.3. Multipoint Conference utilizing Multiple Content Captures	53
12.3.1. Single Media Captures and MCC in the same Advertisement.....	53
12.3.2. Several MCCs in the same Advertisement.....	56
12.3.3. Heterogeneous conference with switching and composition.....	58
12.3.4. Heterogeneous conference with voice activated switching.....	65
13. Acknowledgements.....	67
14. IANA Considerations.....	67
15. Security Considerations.....	68
16. Changes Since Last Version.....	69
17. Authors' Addresses.....	76

1. Introduction

Current telepresence systems, though based on open standards such as RTP [RFC3550] and SIP [RFC3261], cannot easily interoperate with each other. A major factor limiting the interoperability of telepresence systems is the lack of a standardized way to describe and negotiate the use of the multiple streams of audio and video comprising the media flows. This document provides a framework for protocols to enable interoperability by handling multiple streams in a standardized way. The framework is intended to support the use cases described in Use Cases for Telepresence Multistreams [RFC7205] and to meet the requirements in Requirements for Telepresence Multistreams [RFC7262].

The basic session setup for the use cases is based on SIP [RFC3261] and SDP offer/answer [RFC3264]. In addition to basic SIP & SDP offer/answer, CLUE specific signaling is required to exchange the information describing the multiple media streams. The motivation for this framework, an overview of the signaling, and information required to be exchanged is described in subsequent sections of this document. The signaling details and data model are provided in subsequent documents.

2. Terminology

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

3. Definitions

The terms defined below are used throughout this document and companion documents and they are normative. In order to easily identify the use of a defined term, those terms are capitalized.

Advertisement: a CLUE message a Media Provider sends to a Media Consumer describing specific aspects of the content of the media, and any restrictions it has in terms of being able to provide certain Streams simultaneously.

Audio Capture: Media Capture for audio. Denoted as ACn in the examples in this document.

Camera-Left and Right: For Media Captures, camera-left and camera-right are from the point of view of a person observing the rendered media. They are the opposite of Stage-Left and Stage-Right.

Capture: Same as Media Capture.

Capture Device: A device that converts audio and video input into an electrical signal, in most cases to be fed into a media encoder.

Capture Encoding: A specific encoding of a Media Capture, to be sent by a Media Provider to a Media Consumer via RTP.

Capture Scene: a structure representing a spatial region captured by one or more Capture Devices, each capturing media representing a portion of the region. The spatial region represented by a Capture Scene MAY or may not correspond to a real region in physical space, such as a room. A Capture Scene includes attributes and one or more Capture Scene Entries, with each entry including one or more Media Captures.

Capture Scene Entry (CSE): a list of Media Captures of the same media type that together form one way to represent the entire Capture Scene.

Conference: used as defined in [RFC4353], A Framework for Conferencing within the Session Initiation Protocol (SIP).

Configure Message: A CLUE message a Media Consumer sends to a Media Provider specifying which content and media streams it wants to receive, based on the information in a corresponding Advertisement message.

Consumer: short for Media Consumer.

Encoding or Individual Encoding: a set of parameters representing a way to encode a Media Capture to become a Capture Encoding.

Encoding Group: A set of encoding parameters representing a total media encoding capability to be sub-divided across potentially multiple Individual Encodings.

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 rooms to handheld devices.

Front: the portion of the room closest to the cameras. In going towards the back you move away from the cameras.

MCU: Multipoint Control Unit (MCU) - a device that connects two or more endpoints together into one single multimedia conference [RFC5117]. An MCU includes an [RFC4353] like Mixer, without the [RFC4353] requirement to send media to each participant.

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

Media Capture: a source of Media, such as from one or more Capture Devices or constructed from other Media streams.

Media Consumer: an Endpoint or middle box that receives Media streams

Media Provider: an Endpoint or middle box that sends Media streams

Multiple Content Capture: A Capture for audio or video that indicates that the Capture contains multiple audio or video Captures. Single Media Captures may or may not be present in the resultant Capture Encoding depending on time or space. Denoted as MCCn in the example cases in this document.

Plane of Interest: The spatial plane containing the most relevant subject matter.

Provider: Same as Media Provider.

Render: the process of generating a representation from media, such as displayed motion video or sound emitted from loudspeakers.

Simultaneous Transmission Set: a set of Media Captures that can be transmitted simultaneously from a Media Provider.

Single Media Capture: A capture which contains media from a single source capture device, e.g. an audio capture from a single microphone, a video capture from a single camera.

Spatial Relation: The arrangement in space of two objects, in contrast to relation in time or other relationships. See also Camera-Left and Right.

Stage-Left and Right: For Media Captures, Stage-left and Stage-right are the opposite of Camera-left and Camera-right. For the case of a person facing (and captured by) a camera, Stage-left and Stage-right are from the point of view of that person.

Stream: a Capture Encoding sent from a Media Provider to a Media Consumer via RTP [RFC3550].

Stream Characteristics: the media stream attributes commonly used in non-CLUE SIP/SDP environments (such as: media codec, bit rate, resolution, profile/level etc.) as well as CLUE specific attributes, such as the Capture ID or a spatial location.

Video Capture: Media Capture for video. Denoted as VCn in the example cases in this document.

Video Composite: A single image that is formed, normally by an RTP mixer inside an MCU, by combining visual elements from separate sources.

4. Overview & Motivation

This section provides an overview of the functional elements defined in this document to represent a telepresence system. The motivations for the framework described in this document are also provided.

Two key concepts introduced in this document are the terms "Media Provider" and "Media Consumer". A Media Provider represents the entity that sends the media and a Media Consumer represents the entity that receives the media. A Media Provider provides Media in the form of RTP packets, a Media Consumer consumes those RTP packets. Media Providers and Media Consumers can reside in Endpoints or in middleboxes such as Multipoint Control Units (MCUs). A Media Provider in an Endpoint is usually associated with the generation of media for Media Captures; these Media Captures are typically sourced from cameras, microphones, and the like. Similarly, the Media Consumer in an Endpoint is usually associated with renderers, such as screens and loudspeakers. In middleboxes, Media Providers and Consumers can have the form of outputs and inputs, respectively, of RTP mixers, RTP translators, and similar devices. Typically, telepresence devices such as Endpoints and middleboxes would perform as both Media Providers and Media Consumers, the former being concerned with those devices' transmitted media and the latter with those devices' received media. In a few circumstances, a CLUE Endpoint middlebox includes only Consumer or Provider functionality, such as recorder-type Consumers or webcam-type Providers.

The motivations for the framework outlined in this document include the following:

(1) Endpoints in telepresence systems typically have multiple Media Capture and Media Render devices, e.g., multiple cameras and screens. While previous system designs were able to set up calls that would capture media using all cameras and display media on all screens, for example, there was no mechanism that can associate these Media Captures with each other in space and time.

(2) The mere fact that there are multiple capturing and rendering devices, each of which may be configurable in aspects such as zoom,

leads to the difficulty that a variable number of such devices can be used to capture different aspects of a region. The Capture Scene concept allows for the description of multiple setups for those multiple capture devices that could represent sensible operation points of the physical capture devices in a room, chosen by the operator. A Consumer can pick and choose from those configurations based on its rendering abilities and inform the Provider about its choices. Details are provided in section 7.

(3) In some cases, physical limitations or other reasons disallow the concurrent use of a device in more than one setup. For example, the center camera in a typical three-camera conference room can set its zoom objective either to capture only the middle few seats, or all seats of a room, but not both concurrently. The Simultaneous Transmission Set concept allows a Provider to signal such limitations. Simultaneous Transmission Sets are part of the Capture Scene description, and discussed in section 8.

(4) Often, the devices in a room do not have the computational complexity or connectivity to deal with multiple encoding options simultaneously, even if each of these options is sensible in certain scenarios, and even if the simultaneous transmission is also sensible (i.e. in case of multicast media distribution to multiple endpoints). Such constraints can be expressed by the Provider using the Encoding Group concept, described in section 9.

(5) Due to the potentially large number of RTP flows required for a Multimedia Conference involving potentially many Endpoints, each of which can have many Media Captures and media renderers, it has become common to multiplex multiple RTP media flows onto the same transport address, so to avoid using the port number as a multiplexing point and the associated shortcomings such as NAT/firewall traversal. While the actual mapping of those RTP flows to the header fields of the RTP packets is not subject of this specification, the large number of possible permutations of sensible options a Media Provider can make available to a Media Consumer makes a mechanism desirable that allows to narrow down the number of possible options that a SIP offer-answer exchange has to consider. Such information is made available using protocol mechanisms specified in this document and companion documents, although it should be stressed that its use in an implementation is OPTIONAL. Also, there are aspects of the control of both Endpoints and middleboxes/MCUs that dynamically change during the progress of a call, such as audio-level based screen switching, layout changes, and so on, which need to be conveyed. Note that these control

aspects are complementary to those specified in traditional SIP based conference management such as BFCP. An exemplary call flow can be found in section 5.

Finally, all this information needs to be conveyed, and the notion of support for it needs to be established. This is done by the negotiation of a "CLUE channel", a data channel negotiated early during the initiation of a call. An Endpoint or MCU that rejects the establishment of this data channel, by definition, does not support CLUE based mechanisms, whereas an Endpoint or MCU that accepts it is REQUIRED to use it to the extent specified in this document and its companion documents.

5. Overview of the Framework/Model

The CLUE framework specifies how multiple media streams are to be handled in a telepresence conference.

A Media Provider (transmitting Endpoint or MCU) describes specific aspects of the content of the media and the media stream encodings it can send in an Advertisement; and the Media Consumer responds to the Media Provider by specifying which content and media streams it wants to receive in a Configure message. The Provider then transmits the asked-for content in the specified streams.

This Advertisement and Configure typically occur during call initiation but MAY also happen at any time throughout the call, whenever there is a change in what the Consumer wants to receive or (perhaps less common) the Provider can send.

An Endpoint or MCU typically act as both Provider and Consumer at the same time, sending Advertisements and sending Configurations in response to receiving Advertisements. (It is possible to be just one or the other.)

The data model is based around two main concepts: a Capture and an Encoding. A Media Capture (MC), such as audio or video, has attributes to describe the content a Provider can send. Media Captures are described in terms of CLUE-defined attributes, such as spatial relationships and purpose of the capture. Providers tell Consumers which Media Captures they can provide, described in terms of the Media Capture attributes.

A Provider organizes its Media Captures into one or more Capture Scenes, each representing a spatial region, such as a room. A

Consumer chooses which Media Captures it wants to receive from the Capture Scenes.

In addition, the Provider can send the Consumer a description of the Individual Encodings it can send in terms of identifiers which relate to items in SDP.

The Provider can also specify constraints on its ability to provide Media, and a sensible design choice for a Consumer is to take these into account when choosing the content and Capture Encodings it requests in the later offer-answer exchange. Some constraints are due to the physical limitations of devices--for example, a camera may not be able to provide zoom and non-zoom views simultaneously. Other constraints are system based, such as maximum bandwidth.

The following diagram illustrates the information contained in an Advertisement.

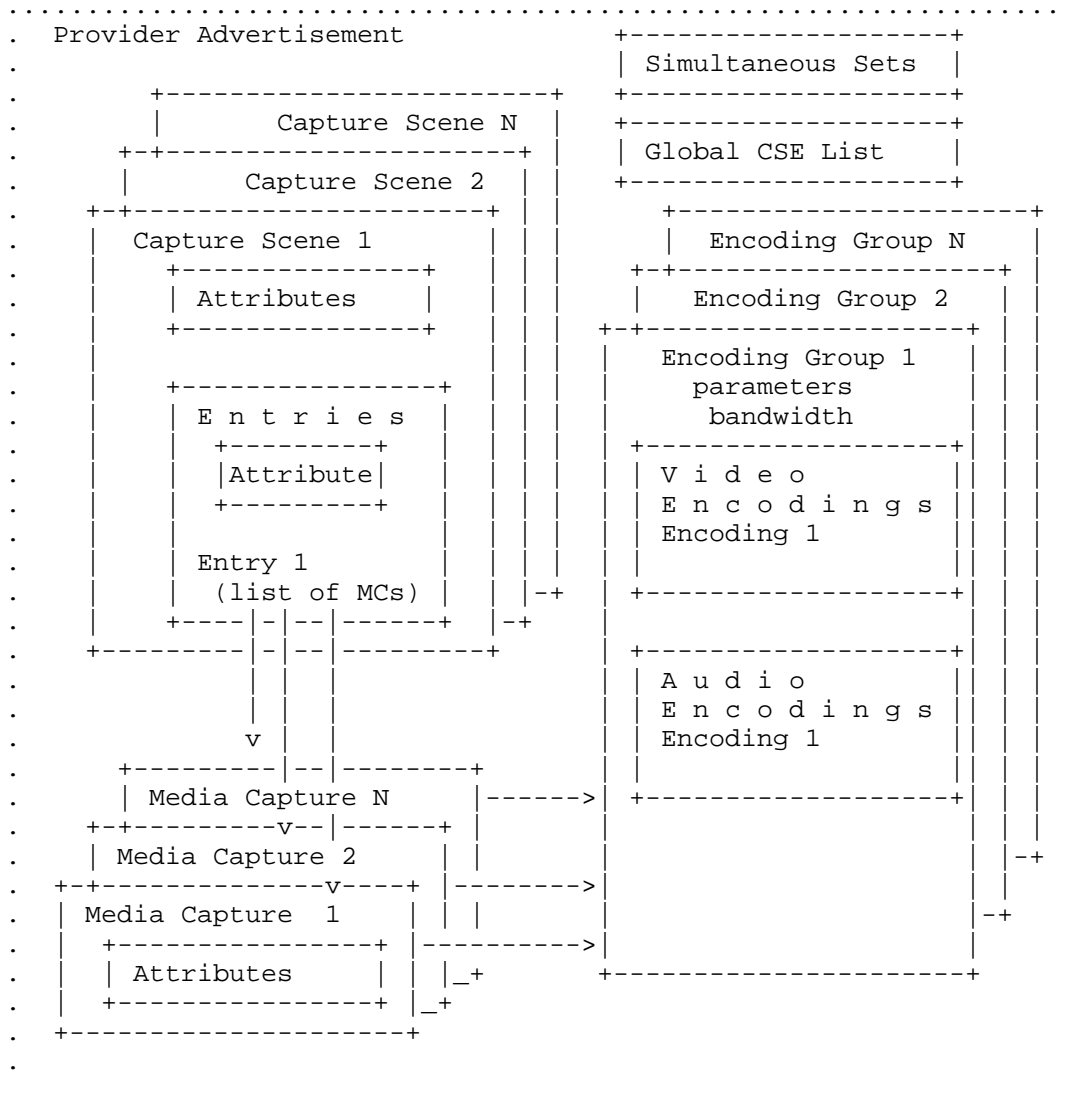


Figure 1: Advertisement Structure

A very brief outline of the call flow used by a simple system (two Endpoints) in compliance with this document can be described as follows, and as shown in the following figure.

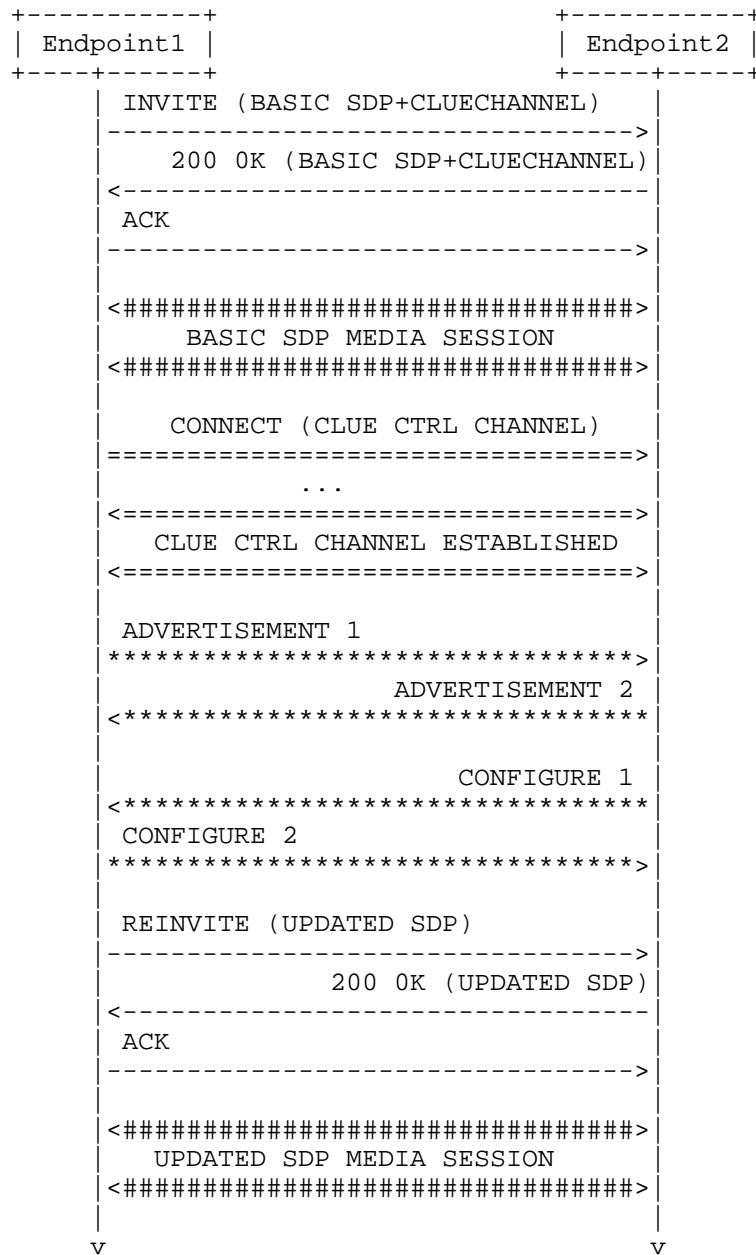


Figure 2: Basic Information Flow

An initial offer/answer exchange establishes a basic media session, for example audio-only, and a CLUE channel between two Endpoints. With the establishment of that channel, the endpoints have consented to use the CLUE protocol mechanisms and, therefore, **MUST** adhere to the CLUE protocol suite as outlined herein.

Over this CLUE channel, the Provider in each Endpoint conveys its characteristics and capabilities by sending an Advertisement as specified herein. The Advertisement is typically not sufficient to set up all media. The Consumer in the Endpoint receives the information provided by the Provider, and can use it for two purposes. First, it **MUST** construct and send a CLUE Configure message to tell the Provider what the Consumer wishes to receive. Second, it **MAY**, but is not necessarily **REQUIRED** to, use the information provided to tailor the SDP it is going to send during the following SIP offer/answer exchange, and its reaction to SDP it receives in that step. It is often a sensible implementation choice to do so, as the representation of the media information conveyed over the CLUE channel can dramatically cut down on the size of SDP messages used in the O/A exchange that follows. Spatial relationships associated with the Media can be included in the Advertisement, and it is often sensible for the Media Consumer to take those spatial relationships into account when tailoring the SDP.

This CLUE exchange **MUST** be followed by an SDP offer answer exchange that not only establishes those aspects of the media that have not been "negotiated" over CLUE, but has also the side effect of setting up the media transmission itself, involving potentially security exchanges, ICE, and whatnot. This step is plain vanilla SIP, with the exception that the SDP used herein, in most (but not necessarily all) cases can be considerably smaller than the SDP a system would typically need to exchange if there were no pre-established knowledge about the Provider and Consumer characteristics. (The need for cutting down SDP size is not quite obvious for a point-to-point call involving simple endpoints; however, when considering a large multipoint conference involving many multi-screen/multi-camera endpoints, each of which can operate using multiple codecs for each camera and microphone, it becomes perhaps somewhat more intuitive.)

During the lifetime of a call, further exchanges **MAY** occur over the CLUE channel. In some cases, those further exchanges lead to a

modified system behavior of Provider or Consumer (or both) without any other protocol activity such as further offer/answer exchanges. For example, voice-activated screen switching, signaled over the CLUE channel, ought not to lead to heavy-handed mechanisms like SIP re-invites. However, in other cases, after the CLUE negotiation an additional offer/answer exchange becomes necessary. For example, if both sides decide to upgrade the call from a single screen to a multi-screen call and more bandwidth is required for the additional video channels compared to what was previously negotiated using offer/answer, a new O/A exchange is REQUIRED.

One aspect of the protocol outlined herein and specified in more detail in companion documents is that it makes available information regarding the Provider's capabilities to deliver Media, and attributes related to that Media such as their spatial relationship, to the Consumer. The operation of the renderer inside the Consumer is unspecified in that it can choose to ignore some information provided by the Provider, and/or not render media streams available from the Provider (although it MUST follow the CLUE protocol and, therefore, MUST gracefully receive and respond (through a Configure) to the Provider's information). All CLUE protocol mechanisms are OPTIONAL in the Consumer in the sense that, while the Consumer MUST be able to receive (and, potentially, gracefully acknowledge) CLUE messages, it is free to ignore the information provided therein.

A CLUE-implementing device interoperates with a device that does not support CLUE, because the non-CLUE device does, by definition, not understand the offer of a CLUE channel in the initial offer/answer exchange and, therefore, will reject it. This rejection MUST be used as the indication to the CLUE-implementing device that the other side of the communication is not compliant with CLUE, and to fall back to behavior that does not require CLUE.

As for the media, Provider and Consumer have an end-to-end communication relationship with respect to (RTP transported) media; and the mechanisms described herein and in companion documents do not change the aspects of setting up those RTP flows and sessions. In other words, the RTP media sessions conform to the negotiated SDP whether or not CLUE is used.

6. Spatial Relationships

In order for a Consumer to perform a proper rendering, it is often necessary or at least helpful for the Consumer to have received

spatial information about the streams it is receiving. CLUE defines a coordinate system that allows Media Providers to describe the spatial relationships of their Media Captures to enable proper scaling and spatially sensible rendering of their streams. The coordinate system is based on a few principles:

- o Simple systems which do not have multiple Media Captures to associate spatially need not use the coordinate model.
- o Coordinates can be either in real, physical units (millimeters), have an unknown scale or have no physical scale. Systems which know their physical dimensions (for example professionally installed Telepresence room systems) MUST always provide those real-world measurements. Systems which don't know specific physical dimensions but still know relative distances MUST use 'unknown scale'. 'No scale' is intended to be used where Media Captures from different devices (with potentially different scales) will be forwarded alongside one another (e.g. in the case of a middle box).
 - * "Millimeters" means the scale is in millimeters.
 - * "Unknown" means the scale is not necessarily millimeters, but the scale is the same for every Capture in the Capture Scene.
 - * "No Scale" means the scale could be different for each capture- an MCU Provider that advertises two adjacent captures and picks sources (which can change quickly) from different endpoints might use this value; the scale could be different and changing for each capture. But the areas of capture still represent a spatial relation between captures.
- o The coordinate system is Cartesian X, Y, Z with the origin at a spatial location of the Provider's choosing. The Provider MUST use the same coordinate system with the same scale and origin for all coordinates within the same Capture Scene.

The direction of increasing coordinate values is:

X increases from Camera-Left to Camera-Right

Y increases from front to back

Z increases from low to high (i.e. floor to ceiling)

7. Media Captures and Capture Scenes

This section describes how Providers can describe the content of media to Consumers.

7.1. Media Captures

Media Captures are the fundamental representations of streams that a device can transmit. What a Media Capture actually represents is flexible:

- o It can represent the immediate output of a physical source (e.g. camera, microphone) or 'synthetic' source (e.g. laptop computer, DVD player).
- o It can represent the output of an audio mixer or video composer
- o It can represent a concept such as 'the loudest speaker'
- o It can represent a conceptual position such as 'the leftmost stream'

To identify and distinguish between multiple Capture instances Captures have a unique identity. For instance: VC1, VC2 and AC1, AC2, where VC1 and VC2 refer to two different video captures and AC1 and AC2 refer to two different audio captures.

Some key points about Media Captures:

- . A Media Capture is of a single media type (e.g. audio or video)
- . A Media Capture is defined in a Capture Scene and is given an advertisement unique identity. The identity may be referenced outside the Capture Scene that defines it through a Multiple Content Capture (MCC)
- . A Media Capture is associated with one or more Capture Scene Entries
- . A Media Capture has exactly one set of spatial information
- . A Media Capture can be the source of one or more Capture Encodings

Each Media Capture can be associated with attributes to describe what it represents.

7.1.1.1. Media Capture Attributes

Media Capture Attributes describe information about the Captures. A Provider can use the Media Capture Attributes to describe the Captures for the benefit of the Consumer of the Advertisement message. Media Capture Attributes include:

- . Spatial information, such as point of capture, point on line of capture, and area of capture, all of which, in combination define the capture field of, for example, a camera;
- . Capture multiplexing information (mono/stereo audio, maximum number of simultaneous encodings per Capture and so on); and
- . Other descriptive information to help the Consumer choose between captures (description, presentation, view, priority, language, person information and type).
- . Control information for use inside the CLUE protocol suite.

The sub-sections below define the Capture attributes.

7.1.1.1.1. Point of Capture

The Point of Capture attribute is a field with a single Cartesian (X, Y, Z) point value which describes the spatial location of the capturing device (such as camera). For an Audio Capture with multiple microphones, the Point of Capture defines the nominal mid-point of the microphones.

7.1.1.1.2. Point on Line of Capture

The Point on Line of Capture attribute is a field with a single Cartesian (X, Y, Z) point value which describes a position in space of a second point on the axis of the capturing device; the first point being the Point of Capture (see above).

Together, the Point of Capture and Point on Line of Capture define an axis of the capturing device, for example the optical axis of a camera or the axis of a microphone. The Media Consumer can use this information to adjust how it renders the received media if it so chooses.

For an Audio Capture, the Media Consumer can use this information along with the Audio Capture Sensitivity Pattern to define a 3-dimensional volume of capture where sounds can be expected to be picked up by the microphone providing this specific audio capture. If the Consumer wants to associate an Audio Capture with a Video

Capture, it can compare this volume with the area of capture for video media to provide a check on whether the audio capture is indeed spatially associated with the video capture. For example, a video area of capture that fails to intersect at all with the audio volume of capture, or is at such a long radial distance from the microphone point of capture that the audio level would be very low, would be inappropriate.

7.1.1.3. Area of Capture

The Area of Capture is a field with a set of four (X, Y, Z) points as a value which describes the spatial location of what is being "captured". By comparing the Area of Capture for different Media Captures within the same Capture Scene a Consumer can determine the spatial relationships between them and render them correctly. Area of Capture does not apply to Audio Captures.

The four points MUST be co-planar, forming a quadrilateral, which defines the Plane of Interest for the particular media capture.

If the Area of Capture is not specified, it means the Media Capture is not spatially related to any other Media Capture.

For a switched capture that switches between different sections within a larger area, the area of capture MUST use coordinates for the larger potential area.

7.1.1.4. Mobility of Capture

The Mobility of Capture attribute indicates whether or not the point of capture, line on point of capture, and area of capture values stay the same over time, or are expected to change (potentially frequently). Possible values are static, dynamic, and highly dynamic.

An example for "dynamic" is a camera mounted on a stand which is occasionally hand-carried and placed at different positions in order to provide the best angle to capture a work task. A camera worn by a person who moves around the room is an example for "highly dynamic". In either case, the effect is that the capture point, capture axis and area of capture change with time.

The capture point of a static capture MUST NOT move for the life of the conference. The capture point of dynamic captures is categorized by a change in position followed by a reasonable period

of stability--in the order of magnitude of minutes. High dynamic captures are categorized by a capture point that is constantly moving. If the "area of capture", "capture point" and "line of capture" attributes are included with dynamic or highly dynamic captures they indicate spatial information at the time of the Advertisement.

7.1.1.5. Audio Capture Sensitivity Pattern

The Audio Capture Sensitivity Pattern attribute applies only to audio captures. This is an optional attribute. This attribute gives information about the nominal sensitivity pattern of the microphone which is the source of the capture. Possible values include patterns such as omni, shotgun, cardioid, hyper-cardioid.

7.1.1.6. Max Capture Encodings

The Max Capture Encodings attribute is an optional attribute indicating the maximum number of Capture Encodings that can be simultaneously active for the Media Capture. The number of simultaneous Capture Encodings is also limited by the restrictions of the Encoding Group for the Media Capture.

7.1.1.7. Description

The Description attribute is a human-readable description (which could be in multiple languages) of the Capture.

7.1.1.8. Presentation

The Presentation attribute indicates that the capture originates from a presentation device, that is one that provides supplementary information to a conference through slides, video, still images, data etc. Where more information is known about the capture it MAY be expanded hierarchically to indicate the different types of presentation media, e.g. presentation.slides, presentation.image etc.

Note: It is expected that a number of keywords will be defined that provide more detail on the type of presentation.

7.1.1.9. View

The View attribute is a field with enumerated values, indicating what type of view the Capture relates to. The Consumer can use this information to help choose which Media Captures it wishes to receive. The value MUST be one of:

Room - Captures the entire scene

Table - Captures the conference table with seated people

Individual - Captures an individual person

Lectern - Captures the region of the lectern including the presenter, for example in a classroom style conference room

Audience - Captures a region showing the audience in a classroom style conference room

7.1.1.10. Language

The language attribute indicates one or more languages used in the content of the Media Capture. Captures MAY be offered in different languages in case of multilingual and/or accessible conferences. A Consumer can use this attribute to differentiate between them and pick the appropriate one.

Note that the Language attribute is defined and meaningful both for audio and video captures. In case of audio captures, the meaning is obvious. For a video capture, "Language" could, for example, be sign interpretation or text.

7.1.1.11. Person Information

The person information attribute allows a Provider to provide specific information regarding the people in a Capture (regardless of whether or not the capture has a Presentation attribute). The Provider may gather the information automatically or manually from a variety of sources however the xCard [RFC6351] format is used to convey the information. This allows various information such as Identification information (section 6.2/[RFC6350]), Communication Information (section 6.4/[RFC6350]) and Organizational information (section 6.6/[RFC6350]) to be communicated. A Consumer may then automatically (i.e. via a policy) or manually select Captures based on information about who is in a Capture. It also allows a

Consumer to render information regarding the people participating in the conference or to use it for further processing.

The Provider may supply a minimal set of information or a larger set of information. However it MUST be compliant to [RFC6350] and supply a "VERSION" and "FN" property. A Provider may supply multiple xCards per Capture of any KIND (section 6.1.4/[RFC6350]).

In order to keep CLUE messages compact the Provider SHOULD use a URI to point to any LOGO, PHOTO or SOUND contained in the xCARD rather than transmitting the LOGO, PHOTO or SOUND data in a CLUE message.

7.1.1.12. Person Type

The person type attribute indicates the type of people contained in the capture in the conference with respect to the meeting agenda (regardless of whether or not the capture has a Presentation attribute). As a capture may include multiple people the attribute may contain multiple values. However values shall not be repeated within the attribute.

An Advertiser associates the person type with an individual capture when it knows that a particular type is in the capture. If an Advertiser cannot link a particular type with some certainty to a capture then it is not included. A Consumer on reception of a capture with a person type attribute knows with some certainty that the capture contains that person type. The capture may contain other person types but the Advertiser has not been able to determine that this is the case.

The types of Captured people include:

- . Chairman - the person responsible for running the conference according to the agenda.
- . Vice-Chairman - the person responsible for assisting the chairman in running the meeting.
- . Minute Taker - the person responsible for recording the minutes of the conference
- . Member - the person has no particular responsibilities with respect to running the meeting.
- . Presenter - the person is scheduled on the agenda to make a presentation in the meeting. Note: This is not related to any "active speaker" functionality.

- . Translator - the person is providing some form of translation or commentary in the meeting.
- . Timekeeper - the person is responsible for maintaining the meeting schedule.

Furthermore the person type attribute may contain one or more strings allowing the Provider to indicate custom meeting specific roles.

7.1.1.13. Priority

The priority attribute indicates a relative priority between different Media Captures. The Provider sets this priority, and the Consumer MAY use the priority to help decide which captures it wishes to receive.

The "priority" attribute is an integer which indicates a relative priority between Captures. For example it is possible to assign a priority between two presentation Captures that would allow a remote endpoint to determine which presentation is more important. Priority is assigned at the individual capture level. It represents the Provider's view of the relative priority between Captures with a priority. The same priority number MAY be used across multiple Captures. It indicates they are equally important. If no priority is assigned no assumptions regarding relative important of the Capture can be assumed.

7.1.1.14. Embedded Text

The Embedded Text attribute indicates that a Capture provides embedded textual information. For example the video Capture MAY contain speech to text information composed with the video image. This attribute is only applicable to video Captures and presentation streams with visual information.

7.1.1.15. Related To

The Related To attribute indicates the Capture contains additional complementary information related to another Capture. The value indicates the identity of the other Capture to which this Capture is providing additional information.

For example, a conference can utilize translators or facilitators that provide an additional audio stream (i.e. a translation or description or commentary of the conference). Where multiple

captures are available, it may be advantageous for a Consumer to select a complementary Capture instead of or in addition to a Capture it relates to.

7.2. Multiple Content Capture

The MCC indicates that one or more Single Media Captures are contained in one Media Capture. Only one Capture type (i.e. audio, video, etc.) is allowed in each MCC instance. The MCC may contain a reference to the Single Media Captures (which may have their own attributes) as well as attributes associated with the MCC itself. A MCC may also contain other MCCs. The MCC MAY reference Captures from within the Capture Scene that defines it or from other Capture Scenes. No ordering is implied by the order that Captures appear within a MCC. A MCC MAY contain no references to other Captures to indicate that the MCC contains content from multiple sources but no information regarding those sources is given.

One or more MCCs may also be specified in a CSE. This allows an Advertiser to indicate that several MCC captures are used to represent a capture scene. Table 14 provides an example of this case.

As outlined in section 7.1. each instance of the MCC has its own Capture identity i.e. MCC1. It allows all the individual captures contained in the MCC to be referenced by a single MCC identity.

The example below shows the use of a Multiple Content Capture:

Capture Scene #1	
VC1	{attributes}
VC2	{attributes}
VCn	{attributes}
MCC1(VC1,VC2,...VCn)	{attributes}
CSE(MCC1)	

Table 1: Multiple Content Capture concept

This indicates that MCC1 is a single capture that contains the Captures VC1, VC2 and VC3 according to any MCC1 attributes.

7.2.1.1. MCC Attributes

Attributes may be associated with the MCC instance and the Single Media Captures that the MCC references. A Provider should avoid providing conflicting attribute values between the MCC and Single Media Captures. Where there is conflict the attributes of the MCC override any that may be present in the individual captures.

A Provider MAY include as much or as little of the original source Capture information as it requires.

There are MCC specific attributes that MUST only be used with Multiple Content Captures. These are described in the sections below. The attributes described in section 7.1.1. MAY also be used with MCCs.

The spatial related attributes of an MCC indicate its area of capture and point of capture within the scene, just like any other media capture. The spatial information does not imply anything about how other captures are composed within an MCC.

For example: A virtual scene could be constructed for the MCC capture with two Video Captures with a "MaxCaptures" attribute set to 2 and an "Area of Capture" attribute provided with an overall area. Each of the individual Captures could then also include an "Area of Capture" attribute with a sub-set of the overall area. The Consumer would then know how each capture is related to others within the scene, but not the relative position of the individual captures within the composed capture.

Capture Scene #1	
VC1	AreaofCapture=(0,0,0)(9,0,0) (0,0,9)(9,0,9)
VC2	AreaofCapture=(10,0,0)(19,0,0) (10,0,9)(19,0,9)
MCC1(VC1,VC2)	MaxCaptures=2 AreaofCapture=(0,0,0)(19,0,0) (0,0,9)(19,0,9)
CSE(MCC1)	

Table 2: Example of MCC and Single Media Capture attributes

The sections below describe the MCC only attributes.

7.2.1.1. Maximum Number of Captures within a MCC

The Maximum Number of Captures MCC attribute indicates the maximum number of individual captures that may appear in a Capture Encoding at a time. The actual number at any given time can be less than this maximum. It may be used to derive how the Single Media Captures within the MCC are composed / switched with regards to space and time.

A Provider can indicate that the number of captures in a MCC capture encoding is equal "=" to the MaxCaptures value or that there may be any number of captures up to and including "<=" the MaxCaptures value. This allows a Provider to distinguish between a MCC that purely represents a composition of sources versus a MCC that represents switched or switched and composed sources.

MaxCaptures MAY be set to one so that only content related to one of the sources are shown in the MCC Capture Encoding at a time or it may be set to any value up to the total number of Source Media Captures in the MCC.

The bullets below describe how the setting of MaxCapture versus the number of captures in the MCC affects how sources appear in a capture encoding:

- . When MaxCaptures is set to <= 1 and the number of captures in the MCC is greater than 1 (or not specified) in the MCC this is a switched case. Zero or 1 captures may be switched into the capture encoding. Note: zero is allowed because of the "<=".
- . When MaxCaptures is set to = 1 and the number of captures in the MCC is greater than 1 (or not specified) in the MCC this is a switched case. Only one capture source is contained in a capture encoding at a time.
- . When MaxCaptures is set to <= N (with N > 1) and the number of captures in the MCC is greater than N (or not specified) this is a switched and composed case. The capture encoding may contain purely switched sources (i.e. <=2 allows for 1 source on its own), or may contain composed and switched sources (i.e. a composition of 2 sources switched between the sources).
- . When MaxCaptures is set to = N (with N > 1) and the number of captures in the MCC is greater than N (or not specified) this

- is a switched and composed case. The capture encoding contains composed and switched sources (i.e. a composition of N sources switched between the sources). It is not possible to have a single source.
- . When MaxCaptures is set to \leq to the number of captures in the MCC this is a switched and composed case. The capture encoding may contain media switched between any number (up to the MaxCaptures) of composed sources.
 - . When MaxCaptures is set to $=$ to the number of captures in the MCC this is a composed case. All the sources are composed into a single capture encoding.

If this attribute is not set then as default it is assumed that all source content can appear concurrently in the Capture Encoding associated with the MCC.

For example: The use of MaxCaptures equal to 1 on a MCC with three Video Captures VC1, VC2 and VC3 would indicate that the Advertiser in the capture encoding would switch between VC1, VC2 or VC3 as there may be only a maximum of one capture at a time.

7.2.1.2. Policy

The Policy MCC Attribute indicates the criteria that the Provider uses to determine when and/or where media content appears in the Capture Encoding related to the MCC.

The attribute is in the form of a token that indicates the policy and index representing an instance of the policy.

The tokens are:

SoundLevel - This indicates that the content of the MCC is determined by a sound level detection algorithm. For example: the loudest (active) speaker is contained in the MCC.

RoundRobin - This indicates that the content of the MCC is determined by a time based algorithm. For example: the Provider provides content from a particular source for a period of time and then provides content from another source and so on.

An index is used to represent an instance in the policy setting. A index of 0 represents the most current instance of the policy, i.e. the active speaker, 1 represents the previous instance, i.e. the previous active speaker and so on.

The following example shows a case where the Provider provides two media streams, one showing the active speaker and a second stream showing the previous speaker.

Capture Scene #1	
VC1	
VC2	
MCC1(VC1,VC2)	Policy=SoundLevel:0 MaxCaptures=1
MCC2(VC1,VC2)	Policy=SoundLevel:1 MaxCaptures=1
CSE(MCC1,MCC2)	

Table 3: Example Policy MCC attribute usage

7.2.1.3. Synchronisation Identity

The Synchronisation Identity MCC attribute indicates how the individual captures in multiple MCC captures are synchronised. To indicate that the Capture Encodings associated with MCCs contain captures from the source at the same time a Provider should set the same Synchronisation Identity on each of the concerned MCCs. It is the Provider that determines what the source for the Captures is, so a Provider can choose how to group together Single Media Captures for the purpose of keeping them synchronized according to the SynchronisationID attribute. For example when the Provider is in an MCU it may determine that each separate CLUE Endpoint is a remote source of media. The Synchronisation Identity may be used across media types, i.e. to synchronize audio and video related MCCs.

Without this attribute it is assumed that multiple MCCs may provide content from different sources at any particular point in time.

For example:

Capture Scene #1	
VC1	Description=Left
VC2	Description=Centre
VC3	Description=Right

AC1 CSE(VC1,VC2,VC3) CSE(AC1)	Description=room
Capture Scene #2	
VC4 VC5 VC6 AC2 CSE(VC4,VC5,VC6) CSE(AC2)	Description=Left Description=Centre Description=Right Description=room
Capture Scene #3	
VC7 AC3	
Capture Scene #4	
VC8 AC4	
Capture Scene #3	
MCC1(VC1,VC4,VC7) MCC2(VC2,VC5,VC8) MCC3(VC3,VC6) MCC4(AC1,AC2,AC3,AC4) CSE(MCC1,MCC2,MCC3) CSE(MCC4)	SynchronisationID=1 MaxCaptures=1 SynchronisationID=1 MaxCaptures=1 MaxCaptures=1 SynchronisationID=1 MaxCaptures=1

Table 4: Example Synchronisation Identity MCC attribute usage

The above Advertisement would indicate that MCC1, MCC2, MCC3 and MCC4 make up a Capture Scene. There would be four capture encodings (one for each MCC). Because MCC1 and MCC2 have the same SynchronisationID, each encoding from MCC1 and MCC2 respectively would together have content from only Capture Scene 1 or only Capture Scene 2 or the combination of VC7 and VC8 at a particular point in time. In this case the Provider has decided the sources

to be synchronized are Scene #1, Scene #2, and Scene #3 and #4 together. The encoding from MCC3 would not be synchronised with MCC1 or MCC2. As MCC4 also has the same Synchronisation Identity as MCC1 and MCC2 the content of the audio encoding will be synchronised with the video content.

7.3. Capture Scene

In order for a Provider's individual Captures to be used effectively by a Consumer, the Provider organizes the Captures into one or more Capture Scenes, with the structure and contents of these Capture Scenes being sent from the Provider to the Consumer in the Advertisement.

A Capture Scene is a structure representing a spatial region containing one or more Capture Devices, each capturing media representing a portion of the region. A Capture Scene includes one or more Capture Scene entries, with each entry including one or more Media Captures. A Capture Scene represents, for example, the video image of a group of people seated next to each other, along with the sound of their voices, which could be represented by some number of VCs and ACs in the Capture Scene Entries. A middle box can also describe in Capture Scenes what it constructs from media Streams it receives.

A Provider MAY advertise one or more Capture Scenes. What constitutes an entire Capture Scene is up to the Provider. A simple Provider might typically use one Capture Scene for participant media (live video from the room cameras) and another Capture Scene for a computer generated presentation. In more complex systems, the use of additional Capture Scenes is also sensible. For example, a classroom may advertise two Capture Scenes involving live video, one including only the camera capturing the instructor (and associated audio), the other including camera(s) capturing students (and associated audio).

A Capture Scene MAY (and typically will) include more than one type of media. For example, a Capture Scene can include several Capture Scene Entries for Video Captures, and several Capture Scene Entries for Audio Captures. A particular Capture MAY be included in more than one Capture Scene Entry.

A Provider MAY express spatial relationships between Captures that are included in the same Capture Scene. However, there is not necessarily the same spatial relationship between Media Captures

that are in different Capture Scenes. In other words, Capture Scenes can use their own spatial measurement system as outlined above in section 6.

A Provider arranges Captures in a Capture Scene to help the Consumer choose which captures it wants to render. The Capture Scene Entries in a Capture Scene are different alternatives the Provider is suggesting for representing the Capture Scene. Each Capture Scene Entry is given an advertisement unique identity. The order of Capture Scene Entries within a Capture Scene has no significance. The Media Consumer can choose to receive all Media Captures from one Capture Scene Entry for each media type (e.g. audio and video), or it can pick and choose Media Captures regardless of how the Provider arranges them in Capture Scene Entries. Different Capture Scene Entries of the same media type are not necessarily mutually exclusive alternatives. Also note that the presence of multiple Capture Scene Entries (with potentially multiple encoding options in each entry) in a given Capture Scene does not necessarily imply that a Provider is able to serve all the associated media simultaneously (although the construction of such an over-rich Capture Scene is probably not sensible in many cases). What a Provider can send simultaneously is determined through the Simultaneous Transmission Set mechanism, described in section 8.

Captures within the same Capture Scene entry MUST be of the same media type - it is not possible to mix audio and video captures in the same Capture Scene Entry, for instance. The Provider MUST be capable of encoding and sending all Captures (that have an encoding group) in a single Capture Scene Entry simultaneously. The order of Captures within a Capture Scene Entry has no significance. A Consumer can decide to receive all the Captures in a single Capture Scene Entry, but a Consumer could also decide to receive just a subset of those captures. A Consumer can also decide to receive Captures from different Capture Scene Entries, all subject to the constraints set by Simultaneous Transmission Sets, as discussed in section 8.

When a Provider advertises a Capture Scene with multiple entries, it is essentially signaling that there are multiple representations of the same Capture Scene available. In some cases, these multiple representations would typically be used simultaneously (for instance a "video entry" and an "audio entry"). In some cases the entries would conceptually be alternatives (for instance an entry consisting of three Video Captures covering the whole room versus

an entry consisting of just a single Video Capture covering only the center of a room). In this latter example, one sensible choice for a Consumer would be to indicate (through its Configure and possibly through an additional offer/answer exchange) the Captures of that Capture Scene Entry that most closely matched the Consumer's number of display devices or screen layout.

The following is an example of 4 potential Capture Scene Entries for an endpoint-style Provider:

1. (VC0, VC1, VC2) - left, center and right camera Video Captures
2. (VC3) - Video Capture associated with loudest room segment
3. (VC4) - Video Capture zoomed out view of all people in the room
4. (AC0) - main audio

The first entry in this Capture Scene example is a list of Video Captures which have a spatial relationship to each other. Determination of the order of these captures (VC0, VC1 and VC2) for rendering purposes is accomplished through use of their Area of Capture attributes. The second entry (VC3) and the third entry (VC4) are alternative representations of the same room's video, which might be better suited to some Consumers' rendering capabilities. The inclusion of the Audio Capture in the same Capture Scene indicates that AC0 is associated with all of those Video Captures, meaning it comes from the same spatial region. Therefore, if audio were to be rendered at all, this audio would be the correct choice irrespective of which Video Captures were chosen.

7.3.1. Capture Scene attributes

Capture Scene Attributes can be applied to Capture Scenes as well as to individual media captures. Attributes specified at this level apply to all constituent Captures. Capture Scene attributes include

- . Human-readable description of the Capture Scene, which could be in multiple languages;
- . xCard scene information
- . Scale information (millimeters, unknown, no scale), as described in Section 6.

7.3.1.1. Scene Information

The Scene information attribute provides information regarding the Capture Scene rather than individual participants. The Provider may gather the information automatically or manually from a variety of sources. The scene information attribute allows a Provider to indicate information such as: organizational or geographic information allowing a Consumer to determine which Capture Scenes are of interest in order to then perform Capture selection. It also allows a Consumer to render information regarding the Scene or to use it for further processing.

As per 7.1.1.11. the xCard format is used to convey this information and the Provider may supply a minimal set of information or a larger set of information.

In order to keep CLUE messages compact the Provider SHOULD use a URI to point to any LOGO, PHOTO or SOUND contained in the xCARD rather than transmitting the LOGO, PHOTO or SOUND data in a CLUE message.

7.3.2. Capture Scene Entry attributes

A Capture Scene can include one or more Capture Scene Entries in addition to the Capture Scene wide attributes described above. Capture Scene Entry attributes apply to the Capture Scene Entry as a whole, i.e. to all Captures that are part of the Capture Scene Entry.

Capture Scene Entry attributes include:

- . Human-readable description (which could be in multiple languages) of the Capture Scene Entry

7.3.3. Global Capture Scene Entry List

An Advertisement can include an optional global Capture Scene Entry list. Each item in this list is a set of one or more Capture Scene Entries of the same media type. Each set of CSEs in the list is a suggestion from the Provider to the Consumer for which CSEs provide a complete representation of the simultaneous captures provided by the Provider, across multiple scenes. The Provider can include multiple sets, to allow a Consumer to choose sets of captures appropriate to its capabilities or application. The choice of how to make these suggestions in the Global CSE list

for what represents all the scenes for which the Provider can send media is up to the Provider. This is very similar to how each CSE represents a particular scene.

As an example, suppose an advertisement has three scenes, and each scene has three CSEs, ranging from one to three video captures in each CSE. The Provider is advertising a total of nine video Captures across three scenes. The Provider can use the Global CSE list to suggest alternatives for Consumers that can't receive all nine video Captures as separate media streams. For accommodating a Consumer that wants to receive three video Captures, a Provider might suggest a single CSE with three Captures and nothing from the other two scenes. Or a Provider might suggest three different CSEs, one from each scene, with a single video Capture in each.

Some additional rules:

- . The ordering of items (sets of CSEs) in the global CSE list is not important.
- . The ordering of CSEs within each set is not important.
- . A particular CSE may be used in multiple sets.
- . The Provider must be capable of encoding and sending all Captures within the CSEs of a given set simultaneously.

8. Simultaneous Transmission Set Constraints

In many practical cases, a Provider has constraints or limitations on its ability to send Captures simultaneously. One type of limitation is caused by the physical limitations of capture mechanisms; these constraints are represented by a simultaneous transmission set. The second type of limitation reflects the encoding resources available, such as bandwidth or video encoding throughput (macroblocks/second). This type of constraint is captured by encoding groups, discussed below.

Some Endpoints or MCUs can send multiple Captures simultaneously; however sometimes there are constraints that limit which Captures can be sent simultaneously with other Captures. A device may not be able to be used in different ways at the same time. Provider Advertisements are made so that the Consumer can choose one of several possible mutually exclusive usages of the device. This type of constraint is expressed in a Simultaneous Transmission Set, which lists all the Captures of a particular media type (e.g. audio, video, text) that can be sent at the same time. There are

different Simultaneous Transmission Sets for each media type in the Advertisement. This is easier to show in an example.

Consider the example of a room system where there are three cameras each of which can send a separate capture covering two persons each- VC0, VC1, VC2. The middle camera can also zoom out (using an optical zoom lens) and show all six persons, VC3. But the middle camera cannot be used in both modes at the same time - it has to either show the space where two participants sit or the whole six seats, but not both at the same time. As a result, VC1 and VC3 cannot be sent simultaneously.

Simultaneous Transmission Sets are expressed as sets of the Media Captures that the Provider could transmit at the same time (though, in some cases, it is not intuitive to do so). If a Multiple Content Capture is included in a Simultaneous Transmission Set it indicates that the Capture Encoding associated with it could be transmitted as the same time as the other Captures within the Simultaneous Transmission Set. It does not imply that the Single Media Captures contained in the Multiple Content Capture could all be transmitted at the same time.

In this example the two simultaneous sets are shown in Table 5. If a Provider advertises one or more mutually exclusive Simultaneous Transmission Sets, then for each media type the Consumer MUST ensure that it chooses Media Captures that lie wholly within one of those Simultaneous Transmission Sets.

+-----+	
Simultaneous Sets	
+-----+	
{VC0, VC1, VC2}	
{VC0, VC3, VC2}	
+-----+	

Table 5: Two Simultaneous Transmission Sets

A Provider OPTIONALLY can include the simultaneous sets in its Advertisement. These simultaneous set constraints apply across all the Capture Scenes in the Advertisement. It is a syntax conformance requirement that the simultaneous transmission sets MUST allow all the media captures in any particular Capture Scene Entry to be used simultaneously. Similarly, the simultaneous transmission sets MUST reflect the simultaneity expressed by any global CSE sets.

For shorthand convenience, a Provider MAY describe a Simultaneous Transmission Set in terms of Capture Scene Entries and Capture Scenes. If a Capture Scene Entry is included in a Simultaneous Transmission Set, then all Media Captures in the Capture Scene Entry are included in the Simultaneous Transmission Set. If a Capture Scene is included in a Simultaneous Transmission Set, then all its Capture Scene Entries (of the corresponding media type) are included in the Simultaneous Transmission Set. The end result reduces to a set of Media Captures in either case.

If an Advertisement does not include Simultaneous Transmission Sets, then the Provider MUST be able to provide all Capture Scenes simultaneously. If multiple capture Scene Entries are in a Capture Scene then the Consumer chooses at most one Capture Scene Entry per Capture Scene for each media type. Likewise, if there are no Simultaneous Transmission Sets and there is a global CSE list, then the Consumer chooses at most one set of CSEs of each media type, from the global CSE list.

If an Advertisement includes multiple Capture Scene Entries in a Capture Scene then the Consumer MAY choose one Capture Scene Entry for each media type, or MAY choose individual Captures based on the Simultaneous Transmission Sets.

9. Encodings

Individual encodings and encoding groups are CLUE's mechanisms allowing a Provider to signal its limitations for sending Captures, or combinations of Captures, to a Consumer. Consumers can map the Captures they want to receive onto the Encodings, with encoding parameters they want. As for the relationship between the CLUE-specified mechanisms based on Encodings and the SIP Offer-Answer exchange, please refer to section 5.

9.1. Individual Encodings

An Individual Encoding represents a way to encode a Media Capture to become a Capture Encoding, to be sent as an encoded media stream from the Provider to the Consumer. An Individual Encoding has a set of parameters characterizing how the media is encoded.

Different media types have different parameters, and different encoding algorithms may have different parameters. An Individual Encoding can be assigned to at most one Capture Encoding at any given time.

Individual Encoding parameters are represented in SDP [RFC4566], not in CLUE messages. For example, for a video encoding using H.26x compression technologies, this can include parameters such as:

- . Maximum bandwidth;
- . Maximum picture size in pixels;
- . Maximum number of pixels to be processed per second;

The bandwidth parameter is the only one that specifically relates to a CLUE Advertisement, as it can be further constrained by the maximum group bandwidth in an Encoding Group.

9.2. Encoding Group

An Encoding Group includes a set of one or more Individual Encodings, and parameters that apply to the group as a whole. By grouping multiple individual Encodings together, an Encoding Group describes additional constraints on bandwidth for the group.

The Encoding Group data structure contains:

- . Maximum bitrate for all encodings in the group combined;
- . A list of identifiers for audio and video encodings, respectively, belonging to the group.

When the Individual Encodings in a group are instantiated into Capture Encodings, each Capture Encoding has a bitrate that **MUST** be less than or equal to the max bitrate for the particular individual encoding. The "maximum bitrate for all encodings in the group" parameter gives the additional restriction that the sum of all the individual capture encoding bitrates **MUST** be less than or equal to the this group value.

The following diagram illustrates one example of the structure of a media Provider's Encoding Groups and their contents.

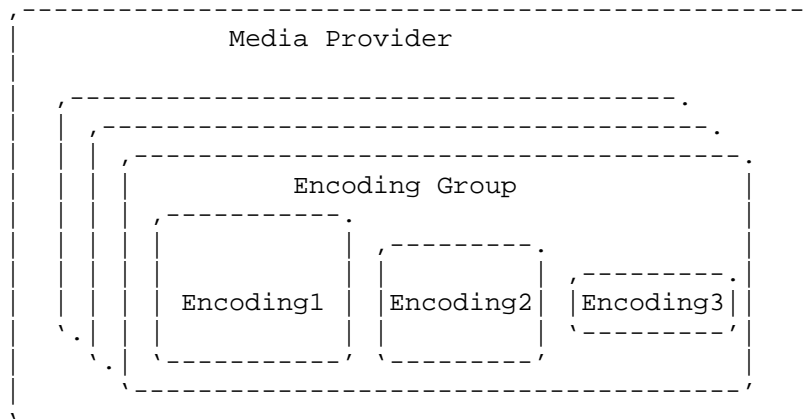


Figure 3: Encoding Group Structure

A Provider advertises one or more Encoding Groups. Each Encoding Group includes one or more Individual Encodings. Each Individual Encoding can represent a different way of encoding media. For example one Individual Encoding may be 1080p60 video, another could be 720p30, with a third being CIF, all in, for example, H.264 format.

While a typical three codec/display system might have one Encoding Group per "codec box" (physical codec, connected to one camera and one screen), there are many possibilities for the number of Encoding Groups a Provider may be able to offer and for the encoding values in each Encoding Group.

There is no requirement for all Encodings within an Encoding Group to be instantiated at the same time.

9.3. Associating Captures with Encoding Groups

Each Media Capture MAY be associated with at least one Encoding Group, which is used to instantiate that Capture into one or more Capture Encodings. Typically MCCs are assigned an Encoding Group and thus become a Capture Encoding. The Captures (including other MCCs) referenced by the MCC do not need to be assigned to an Encoding Group. This means that all the Media Captures referenced by the MCC will appear in the Capture Encoding according to any MCC attributes. This allows an Advertiser to specify Capture attributes associated with the Media Captures without the need to provide an individual Capture Encoding for each of the inputs.

If an Encoding Group is assigned to a Media Capture referenced by the MCC it indicates that this Capture may also have an individual Capture Encoding.

For example:

Capture Scene #1	
VC1	EncodeGroupID=1
VC2	
MCC1(VC1,VC2)	EncodeGroupID=2
CSE(VC1)	
CSE(MCC1)	

Table 6: Example usage of Encoding with MCC and source Captures

This would indicate that VC1 may be sent as its own Capture Encoding from EncodeGroupID=1 or that it may be sent as part of a Capture Encoding from EncodeGroupID=2 along with VC2.

More than one Capture MAY use the same Encoding Group.

The maximum number of streams that can result from a particular Encoding Group constraint is equal to the number of individual Encodings in the group. The actual number of Capture Encodings used at any time MAY be less than this maximum. Any of the Captures that use a particular Encoding Group can be encoded according to any of the Individual Encodings in the group. If there are multiple Individual Encodings in the group, then the Consumer can configure the Provider, via a Configure message, to encode a single Media Capture into multiple different Capture Encodings at the same time, subject to the Max Capture Encodings constraint, with each capture encoding following the constraints of a different Individual Encoding.

It is a protocol conformance requirement that the Encoding Groups MUST allow all the Captures in a particular Capture Scene Entry to be used simultaneously.

10. Consumer's Choice of Streams to Receive from the Provider

After receiving the Provider's Advertisement message (that includes media captures and associated constraints), the Consumer composes

its reply to the Provider in the form of a Configure message. The Consumer is free to use the information in the Advertisement as it chooses, but there are a few obviously sensible design choices, which are outlined below.

If multiple Providers connect to the same Consumer (i.e. in a n MCU-less multiparty call), it is the responsibility of the Consumer to compose Configures for each Provider that both fulfill each Provider's constraints as expressed in the Advertisement, as well as its own capabilities.

In an MCU-based multiparty call, the MCU can logically terminate the Advertisement/Configure negotiation in that it can hide the characteristics of the receiving endpoint and rely on its own capabilities (transcoding/transrating/...) to create Media Streams that can be decoded at the Endpoint Consumers. The timing of an MCU's sending of Advertisements (for its outgoing ports) and Configures (for its incoming ports, in response to Advertisements received there) is up to the MCU and implementation dependent.

As a general outline, a Consumer can choose, based on the Advertisement it has received, which Captures it wishes to receive, and which Individual Encodings it wants the Provider to use to encode the Captures.

On receipt of an Advertisement with an MCC the Consumer treats the MCC as per other non-MCC Captures with the following differences:

- The Consumer would understand that the MCC is a Capture that includes the referenced individual Captures and that these individual Captures are delivered as part of the MCC's Capture Encoding.
- The Consumer may utilise any of the attributes associated with the referenced individual Captures and any Capture Scene attributes from where the individual Captures were defined to choose Captures and for rendering decisions.
- The Consumer may or may not choose to receive all the indicated captures. Therefore it can choose to receive a sub-set of Captures indicated by the MCC.

For example if the Consumer receives:

```
MCC1(VC1,VC2,VC3){attributes}
```

A Consumer could choose all the Captures within a MCCs however if the Consumer determines that it doesn't want VC3 it can return MCC1(VC1,VC2). If it wants all the individual Captures then it returns only the MCC identity (i.e. MCC1). If the MCC in the advertisement does not reference any individual captures, then the Consumer cannot choose what is included in the MCC, it is up to the Provider to decide.

A Configure Message includes a list of Capture Encodings. These are the Capture Encodings the Consumer wishes to receive from the Provider. Each Capture Encoding refers to one Media Capture and one Individual Encoding. A Configure Message does not include references to Capture Scenes or Capture Scene Entries.

For each Capture the Consumer wants to receive, it configures one or more of the Encodings in that Capture's Encoding Group. The Consumer does this by telling the Provider, in its Configure Message, which Encoding to use for each chosen Capture. Upon receipt of this Configure from the Consumer, common knowledge is established between Provider and Consumer regarding sensible choices for the media streams. The setup of the actual media channels, at least in the simplest case, is left to a following offer-answer exchange. Optimized implementations MAY speed up the reaction to the offer-answer exchange by reserving the resources at the time of finalization of the CLUE handshake.

CLUE advertisements and configure messages don't necessarily require a new SDP offer-answer for every CLUE message exchange. But the resulting encodings sent via RTP must conform to the most recent SDP offer-answer result.

In order to meaningfully create and send an initial Configure, the Consumer needs to have received at least one Advertisement from the Provider.

In addition, the Consumer can send a Configure at any time during the call. The Configure MUST be valid according to the most recently received Advertisement. The Consumer can send a Configure either in response to a new Advertisement from the Provider or on its own, for example because of a local change in conditions (people leaving the room, connectivity changes, multipoint related considerations).

When choosing which Media Streams to receive from the Provider, and the encoding characteristics of those Media Streams, the Consumer

advantageously takes several things into account: its local preference, simultaneity restrictions, and encoding limits.

10.1. Local preference

A variety of local factors influence the Consumer's choice of Media Streams to be received from the Provider:

- o if the Consumer is an Endpoint, it is likely that it would choose, where possible, to receive video and audio Captures that match the number of display devices and audio system it has
- o if the Consumer is a middle box such as an MCU, it MAY choose to receive loudest speaker streams (in order to perform its own media composition) and avoid pre-composed video Captures
- o user choice (for instance, selection of a new layout) MAY result in a different set of Captures, or different encoding characteristics, being required by the Consumer

10.2. Physical simultaneity restrictions

Often there are physical simultaneity constraints of the Provider that affect the Provider's ability to simultaneously send all of the captures the Consumer would wish to receive. For instance, a middle box such as an MCU, when connected to a multi-camera room system, might prefer to receive both individual video streams of the people present in the room and an overall view of the room from a single camera. Some Endpoint systems might be able to provide both of these sets of streams simultaneously, whereas others might not (if the overall room view were produced by changing the optical zoom level on the center camera, for instance).

10.3. Encoding and encoding group limits

Each of the Provider's encoding groups has limits on bandwidth and computational complexity, and the constituent potential encodings have limits on the bandwidth, computational complexity, video frame rate, and resolution that can be provided. When choosing the Captures to be received from a Provider, a Consumer device MUST ensure that the encoding characteristics requested for each individual Capture fits within the capability of the encoding it is being configured to use, as well as ensuring that the combined encoding characteristics for Captures fit within the capabilities

of their associated encoding groups. In some cases, this could cause an otherwise "preferred" choice of capture encodings to be passed over in favor of different Capture Encodings--for instance, if a set of three Captures could only be provided at a low resolution then a three screen device could switch to favoring a single, higher quality, Capture Encoding.

11. Extensibility

One important characteristics of the Framework is its extensibility. The standard for interoperability and handling multiple streams must be future-proof. The framework itself is inherently extensible through expanding the data model types. For example:

- o Adding more types of media, such as telemetry, can done by defining additional types of Captures in addition to audio and video.
- o Adding new functionalities, such as 3-D, say, may require additional attributes describing the Captures.

The infrastructure is designed to be extended rather than requiring new infrastructure elements. Extension comes through adding to defined types.

12. Examples - Using the Framework (Informative)

This section gives some examples, first from the point of view of the Provider, then the Consumer, then some multipoint scenarios

12.1. Provider Behavior

This section shows some examples in more detail of how a Provider can use the framework to represent a typical case for telepresence rooms. First an endpoint is illustrated, then an MCU case is shown.

12.1.1. Three screen Endpoint Provider

Consider an Endpoint with the following description:

3 cameras, 3 displays, a 6 person table

- o Each camera can provide one Capture for each 1/3 section of the table
- o A single Capture representing the active speaker can be provided (voice activity based camera selection to a given encoder input port implemented locally in the Endpoint)
- o A single Capture representing the active speaker with the other 2 Captures shown picture in picture within the stream can be provided (again, implemented inside the endpoint)
- o A Capture showing a zoomed out view of all 6 seats in the room can be provided

The audio and video Captures for this Endpoint can be described as follows.

Video Captures:

- o VC0- (the camera-left camera stream), encoding group=EG0, view=table
- o VC1- (the center camera stream), encoding group=EG1, view=table
- o VC2- (the camera-right camera stream), encoding group=EG2, view=table
- o MCC3- (the loudest panel stream), encoding group=EG1, view=table, MaxCaptures=1
- o MCC4- (the loudest panel stream with PiPs), encoding group=EG1, view=room, MaxCaptures=3
- o VC5- (the zoomed out view of all people in the room), encoding group=EG1, view=room
- o VC6- (presentation stream), encoding group=EG1, presentation

The following diagram is a top view of the room with 3 cameras, 3 displays, and 6 seats. Each camera is capturing 2 people. The six seats are not all in a straight line.

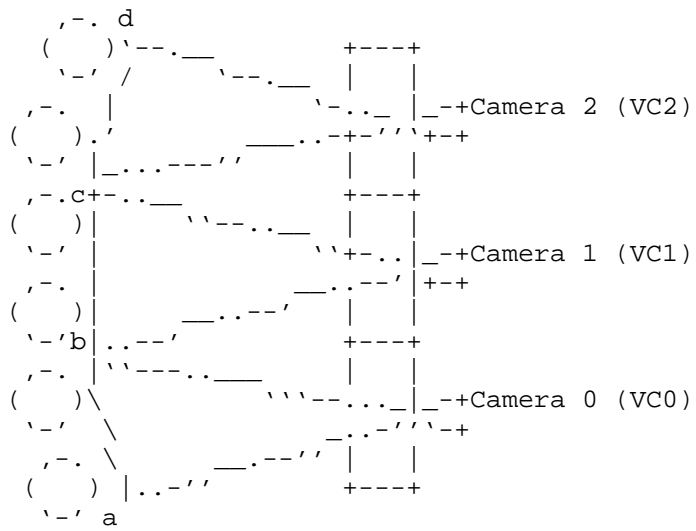


Figure 4: Room Layout

The two points labeled b and c are intended to be at the midpoint between the seating positions, and where the fields of view of the cameras intersect.

The plane of interest for VC0 is a vertical plane that intersects points 'a' and 'b'.

The plane of interest for VC1 intersects points 'b' and 'c'. The plane of interest for VC2 intersects points 'c' and 'd'.

This example uses an area scale of millimeters.

Areas of capture:

	bottom left	bottom right	top left	top right
VC0	(-2011,2850,0)	(-673,3000,0)	(-2011,2850,757)	(-673,3000,757)
VC1	(-673,3000,0)	(673,3000,0)	(-673,3000,757)	(673,3000,757)
VC2	(673,3000,0)	(2011,2850,0)	(673,3000,757)	(2011,3000,757)
MCC3	(-2011,2850,0)	(2011,2850,0)	(-2011,2850,757)	(2011,3000,757)
MCC4	(-2011,2850,0)	(2011,2850,0)	(-2011,2850,757)	(2011,3000,757)
VC5	(-2011,2850,0)	(2011,2850,0)	(-2011,2850,757)	(2011,3000,757)
VC6	none			

Points of capture:

VC0 (-1678,0,800)
VC1 (0,0,800)
VC2 (1678,0,800)
MCC3 none
MCC4 none
VC5 (0,0,800)
VC6 none

In this example, the right edge of the VC0 area lines up with the left edge of the VC1 area. It doesn't have to be this way. There could be a gap or an overlap. One additional thing to note for this example is the distance from a to b is equal to the distance from b to c and the distance from c to d. All these distances are 1346 mm. This is the planar width of each area of capture for VC0, VC1, and VC2.

Note the text in parentheses (e.g. "the camera-left camera stream") is not explicitly part of the model, it is just explanatory text for this example, and is not included in the

model with the media captures and attributes. Also, MCC4 doesn't say anything about how a capture is composed, so the media consumer can't tell based on this capture that MCC4 is composed of a "loudest panel with PiPs".

Audio Captures:

- o AC0 (camera-left), encoding group=EG3
- o AC1 (camera-right), encoding group=EG3
- o AC2 (center) encoding group=EG3
- o AC3 being a simple pre-mixed audio stream from the room (mono), encoding group=EG3
- o AC4 audio stream associated with the presentation video (mono) encoding group=EG3, presentation

Point of capture:	Point on Line of Capture:
AC0 (-1342,2000,800)	(-1342,2925,379)
AC1 (1342,2000,800)	(1342,2925,379)
AC2 (0,2000,800)	(0,3000,379)
AC3 (0,2000,800)	(0,3000,379)
AC4 none	

The physical simultaneity information is:

Simultaneous transmission set #1 {VC0, VC1, VC2, MCC3, MCC4, VC6}

Simultaneous transmission set #2 {VC0, VC2, VC5, VC6}

This constraint indicates it is not possible to use all the VCs at the same time. VC5 cannot be used at the same time as VC1 or MCC3 or MCC4. Also, using every member in the set simultaneously may not make sense - for example MCC3(loudest) and MCC4 (loudest with PIP). (In addition, there are encoding constraints that make choosing all of the VCs in a set impossible. VC1, MCC3, MCC4, VC5, VC6 all use EG1 and EG1 has only 3 ENCs. This constraint shows up in the encoding groups, not in the simultaneous transmission sets.)

In this example there are no restrictions on which audio captures can be sent simultaneously.

Encoding Groups:

This example has three encoding groups associated with the video captures. Each group can have 3 encodings, but with each potential encoding having a progressively lower specification. In this example, 1080p60 transmission is possible (as ENC0 has a maxPps value compatible with that). Significantly, as up to 3 encodings are available per group, it is possible to transmit some video captures simultaneously that are not in the same entry in the capture scene. For example VC1 and MCC3 at the same time.

It is also possible to transmit multiple capture encodings of a single video capture. For example VC0 can be encoded using ENC0 and ENC1 at the same time, as long as the encoding parameters satisfy the constraints of ENC0, ENC1, and EG0, such as one at 4000000 bps and one at 2000000 bps.

```
encodeGroupID=EG0, maxGroupBandwidth=6000000
  encodeID=ENC0, maxWidth=1920, maxHeight=1088, maxFrameRate=60,
    maxPps=124416000, maxBandwidth=4000000
  encodeID=ENC1, maxWidth=1280, maxHeight=720, maxFrameRate=30,
    maxPps=27648000, maxBandwidth=4000000
  encodeID=ENC2, maxWidth=960, maxHeight=544, maxFrameRate=30,
    maxPps=15552000, maxBandwidth=4000000
encodeGroupID=EG1, maxGroupBandwidth=6000000
  encodeID=ENC3, maxWidth=1920, maxHeight=1088, maxFrameRate=60,
    maxPps=124416000, maxBandwidth=4000000
  encodeID=ENC4, maxWidth=1280, maxHeight=720, maxFrameRate=30,
    maxPps=27648000, maxBandwidth=4000000
  encodeID=ENC5, maxWidth=960, maxHeight=544, maxFrameRate=30,
    maxPps=15552000, maxBandwidth=4000000
encodeGroupID=EG2, maxGroupBandwidth=6000000
  encodeID=ENC6, maxWidth=1920, maxHeight=1088, maxFrameRate=60,
    maxPps=124416000, maxBandwidth=4000000
  encodeID=ENC7, maxWidth=1280, maxHeight=720, maxFrameRate=30,
    maxPps=27648000, maxBandwidth=4000000
  encodeID=ENC8, maxWidth=960, maxHeight=544, maxFrameRate=30,
    maxPps=15552000, maxBandwidth=4000000
```

Figure 5: Example Encoding Groups for Video

For audio, there are five potential encodings available, so all five audio captures can be encoded at the same time.

```

encodeGroupID=EG3, maxGroupBandwidth=320000
  encodeID=ENC9, maxBandwidth=64000
  encodeID=ENC10, maxBandwidth=64000
  encodeID=ENC11, maxBandwidth=64000
  encodeID=ENC12, maxBandwidth=64000
  encodeID=ENC13, maxBandwidth=64000

```

Figure 6: Example Encoding Group for Audio

Capture Scenes:

The following table represents the capture scenes for this provider. Recall that a capture scene is composed of alternative capture scene entries covering the same spatial region. Capture Scene #1 is for the main people captures, and Capture Scene #2 is for presentation.

Each row in the table is a separate Capture Scene Entry

-----+ Capture Scene #1 +-----+ VC0, VC1, VC2 MCC3 MCC4 VC5 AC0, AC1, AC2 AC3 +-----+
-----+ Capture Scene #2 +-----+ VC6 AC4 +-----+

Table 7: Example Capture Scene Entries

Different capture scenes are unique to each other, non-overlapping. A consumer can choose an entry from each capture scene. In this case the three captures VC0, VC1, and VC2 are one

way of representing the video from the endpoint. These three captures should appear adjacent next to each other. Alternatively, another way of representing the Capture Scene is with the capture MCC3, which automatically shows the person who is talking. Similarly for the MCC4 and MCC5 alternatives.

As in the video case, the different entries of audio in Capture Scene #1 represent the "same thing", in that one way to receive the audio is with the 3 audio captures (AC0, AC1, AC2), and another way is with the mixed AC3. The Media Consumer can choose an audio capture entry it is capable of receiving.

The spatial ordering is understood by the media capture attributes Area of Capture and Point of Capture and Point on Line of Capture.

A Media Consumer would likely want to choose a capture scene entry to receive based in part on how many streams it can simultaneously receive. A consumer that can receive three people streams would probably prefer to receive the first entry of Capture Scene #1 (VC0, VC1, VC2) and not receive the other entries. A consumer that can receive only one people stream would probably choose one of the other entries.

If the consumer can receive a presentation stream too, it would also choose to receive the only entry from Capture Scene #2 (VC6).

12.1.1.2. Encoding Group Example

This is an example of an encoding group to illustrate how it can express dependencies between encodings.

```
encodeGroupID=EG0 maxGroupBandwidth=6000000
  encodeID=VIDENC0, maxWidth=1920, maxHeight=1088,
    maxFrameRate=60, maxPps=62208000, maxBandwidth=4000000
  encodeID=VIDENC1, maxWidth=1920, maxHeight=1088,
    maxFrameRate=60, maxPps=62208000, maxBandwidth=4000000
  encodeID=AUDENC0, maxBandwidth=96000
  encodeID=AUDENC1, maxBandwidth=96000
  encodeID=AUDENC2, maxBandwidth=96000
```

Here, the encoding group is EG0. Although the encoding group is capable of transmitting up to 6Mbit/s, no individual video encoding can exceed 4Mbit/s.

This encoding group also allows up to 3 audio encodings, AUDENC<0-2>. It is not required that audio and video encodings reside within the same encoding group, but if so then the group's overall maxBandwidth value is a limit on the sum of all audio and video encodings configured by the consumer. A system that does not wish or need to combine bandwidth limitations in this way should instead use separate encoding groups for audio and video in order for the bandwidth limitations on audio and video to not interact.

Audio and video can be expressed in separate encoding groups, as in this illustration.

```

encodeGroupID=EG0 maxGroupBandwidth=6000000
  encodeID=VIDENC0, maxWidth=1920, maxHeight=1088,
    maxFrameRate=60, maxPps=62208000, maxBandwidth=4000000
  encodeID=VIDENC1, maxWidth=1920, maxHeight=1088,
    maxFrameRate=60, maxPps=62208000, maxBandwidth=4000000
encodeGroupID=EG1 maxGroupBandwidth=500000
  encodeID=AUDENC0, maxBandwidth=96000
  encodeID=AUDENC1, maxBandwidth=96000
  encodeID=AUDENC2, maxBandwidth=96000

```

12.1.3. The MCU Case

This section shows how an MCU might express its Capture Scenes, intending to offer different choices for consumers that can handle different numbers of streams. A single audio capture stream is provided for all single and multi-screen configurations that can be associated (e.g. lip-synced) with any combination of video captures at the consumer.

Capture Scene #1	
VC0	VC for a single screen consumer
VC1, VC2	VCs for a two screen consumer
VC3, VC4, VC5	VCs for a three screen consumer
VC6, VC7, VC8, VC9	VCs for a four screen consumer
AC0	AC representing all participants
CSE(VC0)	
CSE(VC1,VC2)	
CSE(VC3,VC4,VC5)	
CSE(VC6,VC7,VC8,VC9)	
CSE(AC0)	

Table 8: MCU main Capture Scenes

If / when a presentation stream becomes active within the conference the MCU might re-advertise the available media as:

Capture Scene #2	note
VC10	video capture for presentation
AC1	presentation audio to accompany VC10
CSE(VC10)	
CSE(AC1)	

Table 9: MCU presentation Capture Scene

12.2. Media Consumer Behavior

This section gives an example of how a Media Consumer might behave when deciding how to request streams from the three screen endpoint described in the previous section.

The receive side of a call needs to balance its requirements, based on number of screens and speakers, its decoding capabilities and available bandwidth, and the provider's capabilities in order to optimally configure the provider's streams. Typically it would want to receive and decode media from each Capture Scene advertised by the Provider.

A sane, basic, algorithm might be for the consumer to go through each Capture Scene in turn and find the collection of Video Captures that best matches the number of screens it has (this might include consideration of screens dedicated to presentation video display rather than "people" video) and then decide between alternative entries in the video Capture Scenes based either on hard-coded preferences or user choice. Once this choice has been made, the consumer would then decide how to configure the provider's encoding groups in order to make best use of the available network bandwidth and its own decoding capabilities.

12.2.1. One screen Media Consumer

MCC3, MCC4 and VC5 are all different entries by themselves, not grouped together in a single entry, so the receiving device should choose between one of those. The choice would come down to

whether to see the greatest number of participants simultaneously at roughly equal precedence (VC5), a switched view of just the loudest region (MCC3) or a switched view with PiPs (MCC4). An endpoint device with a small amount of knowledge of these differences could offer a dynamic choice of these options, in-call, to the user.

12.2.2. Two screen Media Consumer configuring the example

Mixing systems with an even number of screens, "2n", and those with "2n+1" cameras (and vice versa) is always likely to be the problematic case. In this instance, the behavior is likely to be determined by whether a "2 screen" system is really a "2 decoder" system, i.e., whether only one received stream can be displayed per screen or whether more than 2 streams can be received and spread across the available screen area. To enumerate 3 possible behaviors here for the 2 screen system when it learns that the far end is "ideally" expressed via 3 capture streams:

1. Fall back to receiving just a single stream (MCC3, MCC4 or VC5 as per the 1 screen consumer case above) and either leave one screen blank or use it for presentation if / when a presentation becomes active.
2. Receive 3 streams (VC0, VC1 and VC2) and display across 2 screens (either with each capture being scaled to 2/3 of a screen and the center capture being split across 2 screens) or, as would be necessary if there were large bezels on the screens, with each stream being scaled to 1/2 the screen width and height and there being a 4th "blank" panel. This 4th panel could potentially be used for any presentation that became active during the call.
3. Receive 3 streams, decode all 3, and use control information indicating which was the most active to switch between showing the left and center streams (one per screen) and the center and right streams.

For an endpoint capable of all 3 methods of working described above, again it might be appropriate to offer the user the choice of display mode.

12.2.3. Three screen Media Consumer configuring the example

This is the most straightforward case - the Media Consumer would look to identify a set of streams to receive that best matched its available screens and so the VC0 plus VC1 plus VC2 should match optimally. The spatial ordering would give sufficient information for the correct video capture to be shown on the correct screen, and the consumer would either need to divide a single encoding group's capability by 3 to determine what resolution and frame rate to configure the provider with or to configure the individual video captures' encoding groups with what makes most sense (taking into account the receive side decode capabilities, overall call bandwidth, the resolution of the screens plus any user preferences such as motion vs sharpness).

12.3. Multipoint Conference utilizing Multiple Content Captures

The use of MCCs allows the MCU to construct outgoing Advertisements describing complex and media switching and composition scenarios. The following sections provide several examples.

Note: In the examples the identities of the CLUE elements (e.g. Captures, Capture Scene) in the incoming Advertisements overlap. This is because there is no co-ordination between the endpoints. The MCU is responsible for making these unique in the outgoing advertisement.

12.3.1. Single Media Captures and MCC in the same Advertisement

Four endpoints are involved in a Conference where CLUE is used. An MCU acts as a middlebox between the endpoints with a CLUE channel between each endpoint and the MCU. The MCU receives the following Advertisements.

Capture Scene #1	Description=AustralianConfRoom
VC1	Description=Audience
CSE(VC1)	EncodeGroupID=1

Table 10: Advertisement received from Endpoint A

Capture Scene #1	Description=ChinaConfRoom
VC1	Description=Speaker EncodeGroupID=1
VC2	Description=Audience EncodeGroupID=1
CSE(VC1, VC2)	

Table 11: Advertisement received from Endpoint B

Capture Scene #1	Description=USAConfRoom
VC1	Description=Audience EncodeGroupID=1
CSE(VC1)	

Table 12: Advertisement received from Endpoint C

Note: Endpoint B above indicates that it sends two streams.

If the MCU wanted to provide a Multiple Content Capture containing a round robin switched view of the audience from the 3 endpoints and the speaker it could construct the following advertisement:

Advertisement sent to Endpoint F

Capture Scene #1	Description=AustralianConfRoom
VC1 CSE(VC1)	Description=Audience
Capture Scene #2	Description=ChinaConfRoom
VC2 VC3 CSE(VC2, VC3)	Description=Speaker Description=Audience
Capture Scene #3	Description=USAConfRoom
VC4	Description=Audience

CSE(VC4)	
Capture Scene #4	
MCC1(VC1,VC2,VC3,VC4)	Policy=RoundRobin:1 MaxCaptures=1 EncodingGroup=1
CSE(MCC1)	

Table 13: Advertisement sent to Endpoint F - One Encoding

Alternatively if the MCU wanted to provide the speaker as one media stream and the audiences as another it could assign an encoding group to VC2 in Capture Scene 2 and provide a CSE in Capture Scene #4 as per the example below.

Advertisement sent to Endpoint F

Capture Scene #1	Description=AustralianConfRoom
VC1 CSE(VC1)	Description=Audience
Capture Scene #2	Description=ChinaConfRoom
VC2 VC3 CSE(VC2, VC3)	Description=Speaker EncodingGroup=1 Description=Audience
Capture Scene #3	Description=USAConfRoom
VC4 CSE(VC4)	Description=Audience
Capture Scene #4	
MCC1(VC1,VC3,VC4)	Policy=RoundRobin:1 MaxCaptures=1 EncodingGroup=1
MCC2(VC2)	MaxCaptures=1 EncodingGroup=1

CSE2(MCC1,MCC2)	
=====	=====

Table 14: Advertisement sent to Endpoint F - Two Encodings

Therefore a Consumer could choose whether or not to have a separate speaker related stream and could choose which endpoints to see. If it wanted the second stream but not the Australian conference room it could indicate the following captures in the Configure message:

-----	-----
MCC1(VC3,VC4)	Encoding
VC2	Encoding
-----	-----

Table 15: MCU case: Consumer Response

12.3.2. Several MCCs in the same Advertisement

Multiple MCCs can be used where multiple streams are used to carry media from multiple endpoints. For example:

A conference has three endpoints D, E and F. Each end point has three video captures covering the left, middle and right regions of each conference room. The MCU receives the following advertisements from D and E.

-----	-----
Capture Scene #1	Description=AustralianConfRoom
-----	-----
VC1	CaptureArea=Left
VC2	EncodingGroup=1
VC3	CaptureArea=Centre
CSE(VC1,VC2,VC3)	EncodingGroup=1
-----	-----

Table 16: Advertisement received from Endpoint D

-----	-----
Capture Scene #1	Description=ChinaConfRoom
-----	-----
VC1	CaptureArea=Left
	EncodingGroup=1

VC2	CaptureArea=Centre EncodingGroup=1
VC3	CaptureArea=Right EncodingGroup=1
CSE(VC1,VC2,VC3)	

Table 17: Advertisement received from Endpoint E

The MCU wants to offer Endpoint F three Capture Encodings. Each Capture Encoding would contain all the Captures from either Endpoint D or Endpoint E depending based on the active speaker. The MCU sends the following Advertisement:

Capture Scene #1	Description=AustralianConfRoom
VC1	
VC2	
VC3	
CSE(VC1,VC2,VC3)	
Capture Scene #2	Description=ChinaConfRoom
VC4	
VC5	
VC6	
CSE(VC4,VC5,VC6)	
Capture Scene #3	
MCC1(VC1,VC4)	CaptureArea=Left MaxCaptures=1 SynchronisationID=1 EncodingGroup=1
MCC2(VC2,VC5)	CaptureArea=Centre MaxCaptures=1 SynchronisationID=1 EncodingGroup=1
MCC3(VC3,VC6)	CaptureArea=Right MaxCaptures=1 SynchronisationID=1 EncodingGroup=1
CSE(MCC1,MCC2,MCC3)	

Table 17: Advertisement received from Endpoint E

12.3.3. Heterogeneous conference with switching and composition

Consider a conference between endpoints with the following characteristics:

Endpoint A - 4 screens, 3 cameras

Endpoint B - 3 screens, 3 cameras

Endpoint C - 3 screens, 3 cameras

Endpoint D - 3 screens, 3 cameras

Endpoint E - 1 screen, 1 camera

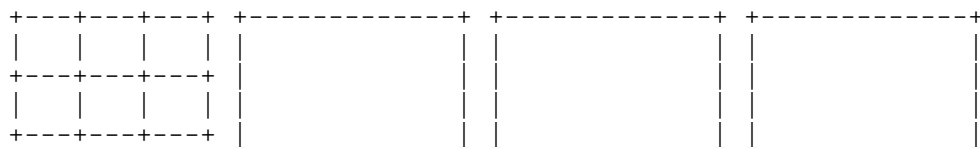
Endpoint F - 2 screens, 1 camera

Endpoint G - 1 screen, 1 camera

This example focuses on what the user in one of the 3-camera multi-screen endpoints sees. Call this person User A, at Endpoint A. There are 4 large display screens at Endpoint A. Whenever somebody at another site is speaking, all the video captures from that endpoint are shown on the large screens. If the talker is at a 3-camera site, then the video from those 3 cameras fills 3 of the screens. If the talker is at a single-camera site, then video from that camera fills one of the screens, while the other screens show video from other single-camera endpoints.

User A hears audio from the 4 loudest talkers.

User A can also see video from other endpoints, in addition to the current talker, although much smaller in size. Endpoint A has 4 screens, so one of those screens shows up to 9 other Media Captures in a tiled fashion. When video from a 3 camera endpoint appears in the tiled area, video from all 3 cameras appears together across the screen with correct spatial relationship among those 3 images.



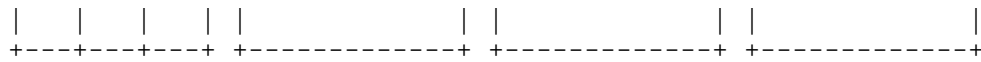


Figure 7: Endpoint A - 4 Screen Display

User B at Endpoint B sees a similar arrangement, except there are only 3 screens, so the 9 other Media Captures are spread out across the bottom of the 3 displays, in a picture-in-picture (PIP) format. When video from a 3 camera endpoint appears in the PIP area, video from all 3 cameras appears together across a single screen with correct spatial relationship.

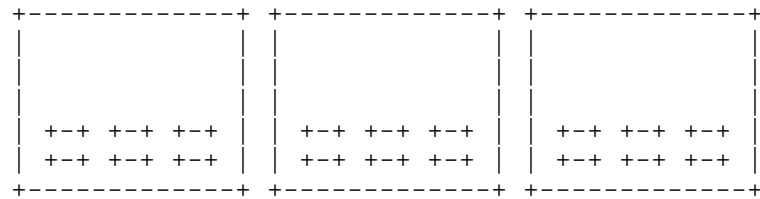


Figure 8: Endpoint B - 3 Screen Display with PiPs

When somebody at a different endpoint becomes the current talker, then User A and User B both see the video from the new talker appear on their large screen area, while the previous talker takes one of the smaller tiled or PIP areas. The person who is the current talker doesn't see themselves; they see the previous talker in their large screen area.

One of the points of this example is that endpoints A and B each want to receive 3 capture encodings for their large display areas, and 9 encodings for their smaller areas. A and B are able to each send the same Configure message to the MCU, and each receive the same conceptual Media Captures from the MCU. The differences are in how they are rendered and are purely a local matter at A and B.

The Advertisements for such a scenario are described below.

Capture Scene #1	Description=Endpoint x
VC1	EncodingGroup=1
VC2	EncodingGroup=1
VC3	EncodingGroup=1
AC1	EncodingGroup=2
CSE1(VC1, VC2, VC3)	

CSE2(AC1)	
-----	-----

Table 19: Advertisement received at the MCU from Endpoints A to D

-----	-----
Capture Scene #1	Description=Endpoint y
-----	-----
VC1	EncodingGroup=1
AC1	EncodingGroup=2
CSE1(VC1)	
CSE2(AC1)	
-----	-----

Table 20: Advertisement received at the MCU from Endpoints E to F

Rather than considering what is displayed the CLUE concentrates more on what the MCU sends. The MCU doesn't know anything about the number of screens an endpoint has.

As Endpoints A to D each advertise that three Captures make up a Capture Scene, the MCU offers these in a "site" switching mode. That is that there are three Multiple Content Captures (and Capture Encodings) each switching between Endpoints. The MCU switches in the applicable media into the stream based on voice activity. Endpoint A will not see a capture from itself.

Using the MCC concept the MCU would send the following Advertisement to endpoint A:

=====	=====
Capture Scene #1	Description=Endpoint B
-----	-----
VC4	Left
VC5	Center
VC6	Right
AC1	
CSE(VC4,VC5,VC6)	
CSE(AC1)	
-----	-----
Capture Scene #2	Description=Endpoint C
-----	-----
VC7	Left
VC8	Center
VC9	Right
-----	-----

AC2 CSE(VC7,VC8,VC9) CSE(AC2)	
Capture Scene #3	Description=Endpoint D
VC10 VC11 VC12 AC3 CSE(VC10,VC11,VC12) CSE(AC3)	Left Center Right
Capture Scene #4	Description=Endpoint E
VC13 AC4 CSE(VC13) CSE(AC4)	
Capture Scene #5	Description=Endpoint F
VC14 AC5 CSE(VC14) CSE(AC5)	
Capture Scene #6	Description=Endpoint G
VC15 AC6 CSE(VC15) CSE(AC6)	

Table 21: Advertisement sent to endpoint A - Source Part

The above part of the Advertisement presents information about the sources to the MCC. The information is effectively the same as the received Advertisements except that there are no Capture Encodings associated with them and the identities have been re-numbered.

In addition to the source Capture information the MCU advertises "site" switching of Endpoints B to G in three streams.

Capture Scene #7	Description=Output3streammix
MCC1(VC4,VC7,VC10,VC13)	CaptureArea=Left MaxCaptures=1 SynchronisationID=1 Policy=SoundLevel:0 EncodingGroup=1
MCC2(VC5,VC8,VC11,VC14)	CaptureArea=Center MaxCaptures=1 SynchronisationID=1 Policy=SoundLevel:0 EncodingGroup=1
MCC3(VC6,VC9,VC12,VC15)	CaptureArea=Right MaxCaptures=1 SynchronisationID=1 Policy=SoundLevel:0 EncodingGroup=1
MCC4() (for audio)	CaptureArea=whole scene MaxCaptures=1 Policy=SoundLevel:0 EncodingGroup=2
MCC5() (for audio)	CaptureArea=whole scene MaxCaptures=1 Policy=SoundLevel:1 EncodingGroup=2
MCC6() (for audio)	CaptureArea=whole scene MaxCaptures=1 Policy=SoundLevel:2 EncodingGroup=2
MCC7() (for audio)	CaptureArea=whole scene MaxCaptures=1 Policy=SoundLevel:3 EncodingGroup=2
CSE(MCC1,MCC2,MCC3) CSE(MCC4,MCC5,MCC6, MCC7)	

Table 22: Advertisement send to endpoint A - switching part

The above part describes the switched 3 main streams that relate to site switching. MaxCaptures=1 indicates that only one Capture from the MCC is sent at a particular time. SynchronisationID=1 indicates that the source sending is synchronised. The provider can choose to group together VC13, VC14, and VC15 for the purpose of switching according to the SynchronisationID. Therefore when the provider switches one of them into an MCC, it can also switch the others even though they are not part of the same Capture Scene.

All the audio for the conference is included in this Scene #7. There isn't necessarily a one to one relation between any audio capture and video capture in this scene. Typically a change in loudest talker will cause the MCU to switch the audio streams more quickly than switching video streams.

The MCU can also supply nine media streams showing the active and previous eight speakers. It includes the following in the Advertisement:

Capture Scene #8	Description=Output9stream
MCC8(VC4,VC5,VC6,VC7,VC8,VC9,VC10,VC11,VC12,VC13,VC14,VC15)	MaxCaptures=1 Policy=SoundLevel:0 EncodingGroup=1
MCC9(VC4,VC5,VC6,VC7,VC8,VC9,VC10,VC11,VC12,VC13,VC14,VC15)	MaxCaptures=1 Policy=SoundLevel:1 EncodingGroup=1
to	to
MCC16(VC4,VC5,VC6,VC7,VC8,VC9,VC10,VC11,VC12,VC13,VC14,VC15)	MaxCaptures=1 Policy=SoundLevel:8 EncodingGroup=1
CSE(MCC8,MCC9,MCC10,MCC11,MCC12,MCC13,MCC14,MCC15,MCC16)	

Table 23: Advertisement sent to endpoint A - 9 switched part

The above part indicates that there are 9 capture encodings. Each of the Capture Encodings may contain any captures from any source site with a maximum of one Capture at a time. Which Capture is present is determined by the policy. The MCCs in this scene do not have any spatial attributes.

Note: The Provider alternatively could provide each of the MCCs above in its own Capture Scene.

If the MCU wanted to provide a composed Capture Encoding containing all of the 9 captures it could Advertise in addition:

Capture Scene #9	Description=NineTiles
MCC13(MCC8,MCC9,MCC10, MCC11,MCC12,MCC13, MCC14,MCC15,MCC16)	MaxCaptures=9 EncodingGroup=1
CSE(MCC13)	

Table 24: Advertisement sent to endpoint A - 9 composed part

As MaxCaptures is 9 it indicates that the capture encoding contains information from 9 sources at a time.

The Advertisement to Endpoint B is identical to the above other than the captures from Endpoint A would be added and the captures from Endpoint B would be removed. Whether the Captures are rendered on a four screen display or a three screen display is up to the Consumer to determine. The Consumer wants to place video captures from the same original source endpoint together, in the correct spatial order, but the MCCs do not have spatial attributes. So the Consumer needs to associate incoming media packets with the original individual captures in the advertisement (such as VC4, VC5, and VC6) in order to know the spatial information it needs for correct placement on the screens.

Editor's note: this is an open issue, about how to associate incoming packets with the original capture that is a constituent of an MCC. This document probably should mention it in an earlier section, after the solution is worked out in the other CLUE documents.

12.3.4. Heterogeneous conference with voice activated switching

This example illustrates how multipoint "voice activated switching" behavior can be realized, with an endpoint making its own decision about which of its outgoing video streams is considered the "active talker" from that endpoint. Then an MCU can decide which is the active talker among the whole conference.

Consider a conference between endpoints with the following characteristics:

Endpoint A - 3 screens, 3 cameras

Endpoint B - 3 screens, 3 cameras

Endpoint C - 1 screen, 1 camera

This example focuses on what the user at endpoint C sees. The user would like to see the video capture of the current talker, without composing it with any other video capture. In this example endpoint C is capable of receiving only a single video stream. The following tables describe advertisements from A and B to the MCU, and from the MCU to C, that can be used to accomplish this.

Capture Scene #1	Description=Endpoint x
VC1	CaptureArea=Left
VC2	EncodingGroup=1 CaptureArea=Center
VC3	EncodingGroup=1 CaptureArea=Right
MCC1(VC1,VC2,VC3)	EncodingGroup=1 MaxCaptures=1 CaptureArea=whole scene Policy=SoundLevel:0
AC1	EncodingGroup=1 CaptureArea=whole scene
CSE1(VC1, VC2, VC3)	EncodingGroup=2
CSE2(MCC1)	
CSE3(AC1)	

Table 25: Advertisement received at the MCU from Endpoints A and B

Endpoints A and B are advertising each individual video capture, and also a switched capture MCC1 which switches between the other three based on who is the active talker. These endpoints do not advertise distinct audio captures associated with each individual video capture, so it would be impossible for the MCU (as a media consumer) to make its own determination of which video capture is the active talker based just on information in the audio streams.

Capture Scene #1	Description=conference
MCC1()	CaptureArea=Left MaxCaptures=1 SynchronisationID=1 Policy=SoundLevel:0 EncodingGroup=1
MCC2()	CaptureArea=Center MaxCaptures=1 SynchronisationID=1 Policy=SoundLevel:0 EncodingGroup=1
MCC3()	CaptureArea=Right MaxCaptures=1 SynchronisationID=1 Policy=SoundLevel:0 EncodingGroup=1
MCC4()	CaptureArea=whole scene MaxCaptures=1 Policy=SoundLevel:0 EncodingGroup=1
MCC5() (for audio)	CaptureArea=whole scene MaxCaptures=1 Policy=SoundLevel:0 EncodingGroup=2
MCC6() (for audio)	CaptureArea=whole scene MaxCaptures=1 Policy=SoundLevel:1 EncodingGroup=2

CSE1(MCC1,MCC2,MCC3	
CSE2(MCC4)	
CSE3(MCC5,MCC6)	

Table 26: Advertisement sent from the MCU to C

The MCU advertises one scene, with four video MCCs. Three of them in CSE1 give a left, center, right view of the conference, with "site switching". MCC4 provides a single video capture representing a view of the whole conference. The MCU intends for MCC4 to be switched between all the other original source captures. In this example advertisement the MCU is not giving all the information about all the other endpoints' scenes and which of those captures is included in the MCCs. The MCU could include all that information if it wants to give the consumers more information, but it is not necessary for this example scenario.

The Provider advertises MCC5 and MCC6 for audio. Both are switched captures, with different SoundLevel policies indicating they are the top two dominant talkers. The Provider advertises CSE3 with both MCCs, suggesting the Consumer should use both if it can.

Endpoint C, in its configure message to the MCU, requests to receive MCC4 for video, and MCC5 and MCC6 for audio. In order for the MCU to get the information it needs to construct MCC4, it has to send configure messages to A and B asking to receive MCC1 from each of them, along with their AC1 audio. Now the MCU can use audio energy information from the two incoming audio streams from A and B to determine which of those alternatives is the current talker. Based on that, the MCU uses either MCC1 from A or MCC1 from B as the source of MCC4 to send to C.

13. Acknowledgements

Allyn Romanow and Brian Baldino were authors of early versions. Mark Gorzynski also contributed much to the initial approach. Many others also contributed, including Christian Groves, Jonathan Lennox, Paul Kyzivat, Rob Hansen, Roni Even, Christer Holmberg, Stephen Botzko, Mary Barnes, John Leslie, Paul Coverdale.

14. IANA Considerations

None.

15. Security Considerations

There are several potential attacks related to telepresence, and specifically the protocols used by CLUE, in the case of conferencing sessions, due to the natural involvement of multiple endpoints and the many, often user-invoked, capabilities provided by the systems.

A middle box involved in a CLUE session can experience many of the same attacks as that of a conferencing system such as that enabled by the XCON framework [RFC 6503]. Examples of attacks include the following: an endpoint attempting to listen to sessions in which it is not authorized to participate, an endpoint attempting to disconnect or mute other users, and theft of service by an endpoint in attempting to create telepresence sessions it is not allowed to create. Thus, it is RECOMMENDED that a middle box implementing the protocols necessary to support CLUE, follow the security recommendations specified in the conference control protocol documents. In the case of CLUE, SIP is the default conferencing protocol, thus the security considerations in RFC 4579 MUST be followed.

One primary security concern, surrounding the CLUE framework introduced in this document, involves securing the actual protocols and the associated authorization mechanisms. These concerns apply to endpoint to endpoint sessions, as well as sessions involving multiple endpoints and middle boxes. Figure 2 in section 5 provides a basic flow of information exchange for CLUE and the protocols involved.

As described in section 5, CLUE uses SIP/SDP to establish the session prior to exchanging any CLUE specific information. Thus the security mechanisms recommended for SIP [RFC 3261], including user authentication and authorization, SHOULD be followed. In addition, the media is based on RTP and thus existing RTP security mechanisms, such as DTLS/SRTP, MUST be supported.

A separate data channel is established to transport the CLUE protocol messages. The contents of the CLUE protocol messages are based on information introduced in this document, which is represented by an XML schema for this information defined in the CLUE data model [ref]. Some of the information which could possibly introduce privacy concerns is the xCard information as described in section x. In addition, the (text) description field in the Media Capture attribute (section 7.1.1.7) could possibly

reveal sensitive information or specific identities. The same would be true for the descriptions in the Capture Scene (section 7.3.1) and Capture Scene Entry (7.3.2) attributes. One other important consideration for the information in the xCard as well as the description field in the Media Capture and Capture Scene Entry attributes is that while the endpoints involved in the session have been authenticated, there is no assurance that the information in the xCard or description fields is authentic. Thus, this information SHOULD not be used to make any authorization decisions and the participants in the sessions SHOULD be made aware of this.

While other information in the CLUE protocol messages does not reveal specific identities, it can reveal characteristics and capabilities of the endpoints. That information could possibly uniquely identify specific endpoints. It might also be possible for an attacker to manipulate the information and disrupt the CLUE sessions. It would also be possible to mount a DoS attack on the CLUE endpoints if a malicious agent has access to the data channel. Thus, It MUST be possible for the endpoints to establish a channel which is secure against both message recovery and message modification. Further details on this are provided in the CLUE data channel solution document.

There are also security issues associated with the authorization to perform actions at the CLUE endpoints to invoke specific capabilities (e.g., re-arranging screens, sharing content, etc.). However, the policies and security associated with these actions are outside the scope of this document and the overall CLUE solution.

16. Changes Since Last Version

NOTE TO THE RFC-Editor: Please remove this section prior to publication as an RFC.

Changes from 15 to 16:

1. Remove Audio Channel Format attribute
2. Add Audio Capture Sensitivity Pattern attribute
3. Clarify audio spatial information regarding point of capture and point on line of capture. Area of capture does not apply to audio.

4. Update section 12 example for new treatment of audio spatial information.
5. Clean up wording of some definitions, and various places in sections 5 and 10.
6. Remove individual encoding parameter paragraph from section 9.
7. Update Advertisement diagram.
8. Update Acknowledgements.
9. References to use cases and requirements now refer to RFCs.
10. Minor editorial changes.

Changes from 14 to 15:

1. Add "=" and "<=" qualifiers to MaxCaptures attribute, and clarify the meaning regarding switched and composed MCC.
2. Add section 7.3.3 Global Capture Scene Entry List, and a few other sentences elsewhere that refer to global CSE sets.
3. Clarify: The Provider MUST be capable of encoding and sending all Captures (*that have an encoding group*) in a single Capture Scene Entry simultaneously.
4. Add voice activated switching example in section 12.
5. Change name of attributes Participant Info/Type to Person Info/Type.
6. Clarify the Person Info/Type attributes have the same meaning regardless of whether or not the capture has a Presentation attribute.
7. Update example section 12.1 to be consistent with the rest of the document, regarding MCC and capture attributes.
8. State explicitly each CSE has a unique ID.

Changes from 13 to 14:

1. Fill in section for Security Considerations.
2. Replace Role placeholder with Participant Information, Participant Type, and Scene Information attributes.
3. Spatial information implies nothing about how constituent media captures are combined into a composed MCC.
4. Clean up MCC example in Section 12.3.3. Clarify behavior of tiled and PIP display windows. Add audio. Add new open issue about associating incoming packets to original source capture.
5. Remove editor's note and associated statement about RTP multiplexing at end of section 5.
6. Remove editor's note and associated paragraph about overloading media channel with both CLUE and non-CLUE usage, in section 5.
7. In section 10, clarify intent of media encodings conforming to SDP, even with multiple CLUE message exchanges. Remove associated editor's note.

Changes from 12 to 13:

1. Added the MCC concept including updates to existing sections to incorporate the MCC concept. New MCC attributes: MaxCaptures, SynchronisationID and Policy.
2. Removed the "composed" and "switched" Capture attributes due to overlap with the MCC concept.
3. Removed the "Scene-switch-policy" CSE attribute, replaced by MCC and SynchronisationID.
4. Editorial enhancements including numbering of the Capture attribute sections, tables, figures etc.

Changes from 11 to 12:

1. Ticket #44. Remove note questioning about requiring a Consumer to send a Configure after receiving Advertisement.

2. Ticket #43. Remove ability for consumer to choose value of attribute for scene-switch-policy.
3. Ticket #36. Remove computational complexity parameter, MaxGroupPps, from Encoding Groups.
4. Reword the Abstract and parts of sections 1 and 4 (now 5) based on Mary's suggestions as discussed on the list. Move part of the Introduction into a new section Overview & Motivation.
5. Add diagram of an Advertisement, in the Overview of the Framework/Model section.
6. Change Intended Status to Standards Track.
7. Clean up RFC2119 keyword language.

Changes from 10 to 11:

1. Add description attribute to Media Capture and Capture Scene Entry.
2. Remove contradiction and change the note about open issue regarding always responding to Advertisement with a Configure message.
3. Update example section, to cleanup formatting and make the media capture attributes and encoding parameters consistent with the rest of the document.

Changes from 09 to 10:

1. Several minor clarifications such as about SDP usage, Media Captures, Configure message.
2. Simultaneous Set can be expressed in terms of Capture Scene and Capture Scene Entry.
3. Removed Area of Scene attribute.
4. Add attributes from draft-groves-clue-capture-attr-01.

5. Move some of the Media Capture attribute descriptions back into this document, but try to leave detailed syntax to the data model. Remove the OUTSOURCE sections, which are already incorporated into the data model document.

Changes from 08 to 09:

1. Use "document" instead of "memo".
2. Add basic call flow sequence diagram to introduction.
3. Add definitions for Advertisement and Configure messages.
4. Add definitions for Capture and Provider.
5. Update definition of Capture Scene.
6. Update definition of Individual Encoding.
7. Shorten definition of Media Capture and add key points in the Media Captures section.
8. Reword a bit about capture scenes in overview.
9. Reword about labeling Media Captures.
10. Remove the Consumer Capability message.
11. New example section heading for media provider behavior
12. Clarifications in the Capture Scene section.
13. Clarifications in the Simultaneous Transmission Set section.
14. Capitalize defined terms.
15. Move call flow example from introduction to overview section
16. General editorial cleanup
17. Add some editors' notes requesting input on issues

18. Summarize some sections, and propose details be outsourced to other documents.

Changes from 06 to 07:

1. Ticket #9. Rename Axis of Capture Point attribute to Point on Line of Capture. Clarify the description of this attribute.
2. Ticket #17. Add "capture encoding" definition. Use this new term throughout document as appropriate, replacing some usage of the terms "stream" and "encoding".
3. Ticket #18. Add Max Capture Encodings media capture attribute.
4. Add clarification that different capture scene entries are not necessarily mutually exclusive.

Changes from 05 to 06:

1. Capture scene description attribute is a list of text strings, each in a different language, rather than just a single string.
2. Add new Axis of Capture Point attribute.
3. Remove appendices A.1 through A.6.
4. Clarify that the provider must use the same coordinate system with same scale and origin for all coordinates within the same capture scene.

Changes from 04 to 05:

1. Clarify limitations of "composed" attribute.
2. Add new section "capture scene entry attributes" and add the attribute "scene-switch-policy".
3. Add capture scene description attribute and description language attribute.
4. Editorial changes to examples section for consistency with the rest of the document.

Changes from 03 to 04:

1. Remove sentence from overview - "This constitutes a significant change ..."
2. Clarify a consumer can choose a subset of captures from a capture scene entry or a simultaneous set (in section "capture scene" and "consumer's choice...").
3. Reword first paragraph of Media Capture Attributes section.
4. Clarify a stereo audio capture is different from two mono audio captures (description of audio channel format attribute).
5. Clarify what it means when coordinate information is not specified for area of capture, point of capture, area of scene.
6. Change the term "producer" to "provider" to be consistent (it was just in two places).
7. Change name of "purpose" attribute to "content" and refer to RFC4796 for values.
8. Clarify simultaneous sets are part of a provider advertisement, and apply across all capture scenes in the advertisement.
9. Remove sentence about lip-sync between all media captures in a capture scene.
10. Combine the concepts of "capture scene" and "capture set" into a single concept, using the term "capture scene" to replace the previous term "capture set", and eliminating the original separate capture scene concept.

Informative References

Edt. Note: Decide which of these really are Normative References.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E.

- Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3264] Rosenberg, J., Schulzrinne, H., "An Offer/Answer Model with the Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC4353] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353, February 2006.
- [RFC4579] Johnston, A., Levin, O., "SIP Call Control - Conferencing for User Agents", RFC 4579, August 2006
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.
- [RFC7205] Romanow, A., Botzko, S., Duckworth, M., Even, R., "Use Cases for Telepresence Multistreams", RFC 7205, April 2014.
- [RFC7262] Romanow, A., Botzko, S., Barnes, M., "Requirements for Telepresence Multistreams", RFC 7262, June 2014.

17. Authors' Addresses

Mark Duckworth (editor)
Polycom
Andover, MA 01810
USA

Email: mark.duckworth@polycom.com

Andrew Pepperell
Acano
Uxbridge, England
UK

Email: apeppere@gmail.com

Stephan Wenger
Vidyo, Inc.
433 Hackensack Ave.
Hackensack, N.J. 07601
USA

Email: stewe@stewe.org

CLUE Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 26, 2014

R. Presta
S. Romano
University of Napoli
June 24, 2014

CLUE protocol
draft-ietf-clue-protocol-01

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 [I-D.ietf-clue-telepresence-requirements]. The companion document [I-D.kyzivat-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 December 26, 2014.

Copyright Notice

Copyright (c) 2014 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. Overview of the CLUE protocol	4
4. Protocol messages	6
4.1. OPTIONS	8
4.2. OPTIONS RESPONSE	10
4.3. ADVERTISEMENT	11
4.4. ADVERTISEMENT ACKNOWLEDGEMENT	12
4.5. CONFIGURE	13
4.6. CONFIGURE RESPONSE	14
4.7. READV	14
4.8. READV RESPONSE	15
4.9. Response codes and reason strings	16
5. Protocol state machines	18
6. CLUE Participant's state machine	18
6.1. Media Consumer's state machine	21
6.2. Media Provider's state machine	23
7. Versioning	25
8. Extensions and options	26
9. XML Schema	28
10. ADVERTISEMENT examples	33
10.1. Simple ADV	34
10.2. ADV with MCCs	39
11. Diff with draft-ietf-clue-protocol-00	46
12. Diff with draft-presta-clue-protocol-04	47
13. Diff with draft-presta-clue-protocol-03	47
14. Diff with draft-presta-clue-protocol-02	47
15. Acknowledgments	47
16. Informative References	48

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 fully 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 to 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 depicted respectively in [I-D.kyzivat-clue-signaling] and [I-D.kyzivat-clue-signaling]. CLUE protocol messages flowing upon such channel are detailed in the following, 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 [I-D.ietf-clue-telepresence-requirements]. We briefly recall herein some of the main terms exploited in the document. We further introduce the definition of CLUE Participant.

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

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 a MCU) able to send Media Streams.

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

3. Overview of the CLUE protocol

The CLUE protocol has been conceived to enable CLUE telepresence session. It is designed in order to address SDP limitations in terms of the description of several 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 the information about the features of the flowing multimedia streams that is needed to enable a "being there" rendering. 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 that enables 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 reliable and ordered SCTP over DTLS transport channel connecting two CLUE Participants. Messages carries 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 ends up 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. 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 offer-answer 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 integer, 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 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 is 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 and 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, the CONFIGURE, and the READV RESPONSE messages), in order to use the concept defined in [I-D.ietf-clue-framework].

The CLUE protocol messages that have been defined up to now are the following:

- o OPTIONS
- o OPTIONS RESPONSE
- o ADVERTISEMENT (ADV)
- o ADVERTISEMENT ACKNOWLEDGE (ACK)

- o CONFIGURE (CONF)
- o CONFIGURE RESPONSE
- o READV
- o READV 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 figure:

```
<!-- 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="xs:string" use="required"/>
</xs:complexType>
```

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", and the like.

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="xs:string" 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="xs:string" 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. For example, if the "v" attribute has a value of "3.4" and there is not a <supportedVersions> tag in the OPTIONS message, it means the CI supports only major version 3 with all the minor versions comprised between 3.0 the 3.4 included. If a <supportedVersion> is provided, at least one <version> tag MUST be included.

The <supportedOptions> element specifies the list of the options supported by the CI. If there is no <supportedOptions> in the OPTIONS message, the CI does not support anything more 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. [to be discussed: difference between options and extensions]

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 (2 WAY) -->
<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="xs:string" 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>
```

After the reception of such 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 will 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

This message 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. 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>) and of the global capture entries (>globalCaptureEntries>), and the list of the represented participants (>participants>). 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 fields -->
        <xs:element name="mediaCaptures" type="dm:mediaCapturesType"/>
        <xs:element name="encodingGroups" type="dm:encodingGroupsType"/>
        <xs:element name="captureScenes" type="dm:captureScenesType"/>
        <xs:element name="simultaneousSets" type="dm:simultaneousSetsType"
          minOccurs="0"/>
        <xs:element name="globalCaptureEntries" type="dm:globalCaptureEntriesType"
          minOccurs="0"/>
        <xs:element name="participants" type="dm:participantsType" minOccurs="0"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

[to be discussed: a "delta" mechanism for advertising only the changes with respect to the previous notification should be adopted. Similar approaches have been proposed for partial notifications in centralized conferencing frameworks ([RFC6502]), leveraging the XML diff codification mechanism defined in [RFC5261]].

4.4. ADVERTISEMENT ACKNOWLEDGEMENT

The ACK message 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 CLUE CONFIGURE message 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>
```

In the >advSequenceNr< element is contained the sequence number of the ADVERTISEMENT or of the READV RESPONSE message the CONFIGURE 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 ADV ACK message has been already sent back to the MP and if the CONFIGURE refers to a READV RESPONSE message.

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

4.6. CONFIGURE RESPONSE

```
<!-- CONFIGURE RESPONSE MESSAGE TYPE -->
<xs:complexType name="configureResponseType">
  <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:integer"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The CONF RESPONSE message 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 number of the CONF message the response refers to.

4.7. READV

The READV message is a request the MC issues to the MP to retrieve an updated version of the MP's telepresence offer. The content of the READV message is specified in the following.


```
<!-- CLUE READV MESSAGE TYPE -->
<xs:complexType name="readvMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <xs:element name="lastReceivedAdv" type="xs:short"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The <lastReceivedAdv> element specifies the sequence number of the last ADVERTISEMENT or READV RESPONSE correctly received by the MC.

4.8. READV RESPONSE

The READV RESPONSE is sent by the MP to the MC to reply to a READV message. As shown in the schema below, it contains, besides a response code and a reason string, all the information carried within an ADVERTISEMENT message (media captures, encoding groups, and so on). If there are no updates with respect to the last telepresence offer successfully delivered to the MC (i.e., that having the sequence number specified in the <lastReceiveAdv> field of the READV message), the READV RESPONSE SHOULD carry only the response code with the reason string.

```
<!-- CLUE READV RESPONSE MESSAGE TYPE -->
<xs:complexType name="readvResponseMessageType">
  <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="readvSequenceNr" type="xs:string" minOccurs="0"/>
        <xs:element name="mediaCaptures" type="dm:mediaCapturesType" minOccurs="0"/>
        <xs:element name="encodingGroups" type="dm:encodingGroupsType" minOccurs="0"/>
        <xs:element name="captureScenes" type="dm:captureScenesType" minOccurs="0"/>
        <xs:element name="simultaneousSets" type="dm:simultaneousSetsType" minOccurs="0"/>
        <xs:element name="globalCaptureEntries" type="dm:globalCaptureEntriesType" minOccurs="0"/>
        <xs:element name="participants" type="dm:participantsType" minOccurs="0"/>
        <xs:any namespace="##other"
          processContents="lax" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

4.9. Response codes and reason strings

Examples of response codes and strings are provided in the following table. Response codes can be designed by adhering to the HTTP semantics, as shown below.

Response code	Response string	Description
410	Bad syntax	The XML syntax of the CONF message is not correct.
411	Invalid value	The CONF message contains an invalid parameter value.
412	Invalid identifier	The identifier used for requesting a capture is not valid or unknown.
413	Conflicting values	The CONF message contains values that cannot be used together.
420	Invalid sequencing	The sequence number of the CONF message is out of date or corresponds to an obsoleted ADV.
510	Version not supported	The CLUE protocol version of the CONF message is not supported by the MP.
511	Option not supported	The option requested in the CONF message is not supported by the MP.

... TBC.

Response code family	Description
1XX	Temporary info
2XX	Success
3XX	Redirection
4XX	Client error
5XX	Server error

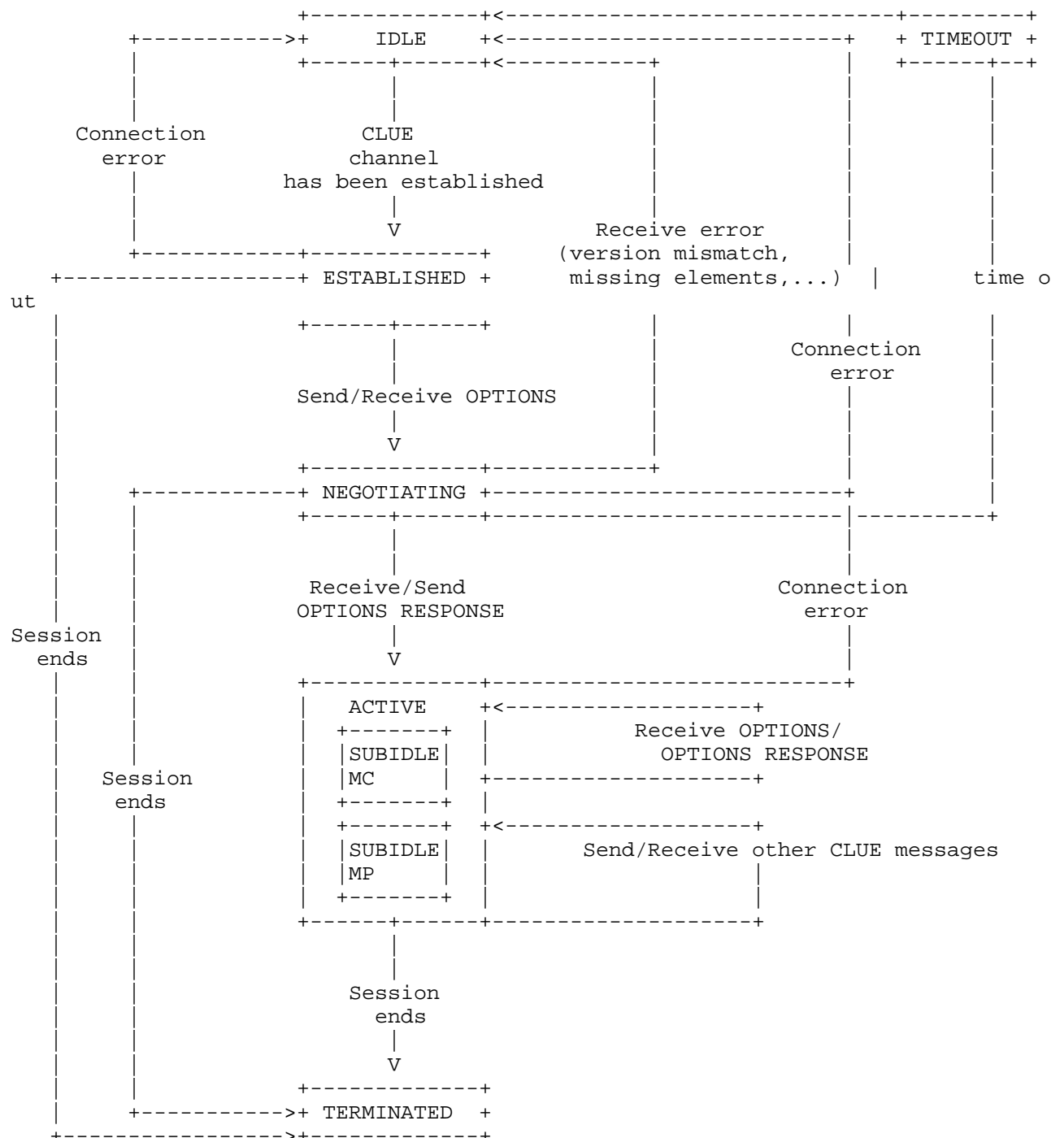
5. Protocol state machines

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.kyzivat-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 MC process and with the MP process.

6. CLUE Participant's state machine

The main state machines focus on describing the states of CLUE channel from a CLUE channel initiator/receiver. In the IDLE state, when the CP has established a CLUE channel, the main state moves to the ESTABLISHED state. When in the ESTABLISHED state, if the CP is the Channel Initiator (CI), it prepares sending an OPTIONS message for version negotiation; otherwise, if the is the Channel Receiver

(CR), it listens to the channel for an OPTIONS message for version negotiation. If an OPTIONS message is sent or is received, the CP moves to the NEGOTIATING state. If the CP checks some error in the request message received, the main state goes back to the IDLE state. [TODO: check this] When in the NEGOTIATING state, the CR prepares an OPTIONS RESPONSE message while the CI listens to the channel for an OPTIONS RESPONSE. If an OPTIONS RESPONSE message for version negotiation is sent or is received, the main state moves to the ACTIVE state. If the CI checks some error in the OPTIONS RESPONSE message received or receives an OPTIONS RESPONSE indicating an error, it goes back to the IDLE state. When the CP enters in the ACTIVE state, it creates two sub state machines which are the MC state machine and the MP state machine, accordingly to the supported roles. When in the ACTIVE state, if the CP receives a further OPTIONS message for version negotiation or a further OPTIONS RESPONSE messages for version negotiation, it MUST ignore the messages and keep in the ACTIVE state. When in the ACTIVE state, the CP delegates the sending and the processing of the CLUE messages the appropriate MP/MC sub-state machines. The TERMINATED state is reachable from each of the aforementioned states whenever the session is canceled or released. The IDLE state is reachable from each of the aforementioned states whenever the underlying channel is closed due to connection error. [TODO: CLUE messages to cancel/release the session] [TODO: check the diagram]



6.1. Media Consumer's state machine

An MC in the WAIT FOR ADV state is waiting for an ADV coming from the MP. If the timeout expires ("timeout"), the MC switches to the TIMEOUT state.

In the TIMEOUT state, if the number of trials is below the retry threshold, the MC sends a READV message to the MP ("send RE-ADV"), switching back to the WAIT FOR ADV. Otherwise, the MC moves to the TERMINATED state.

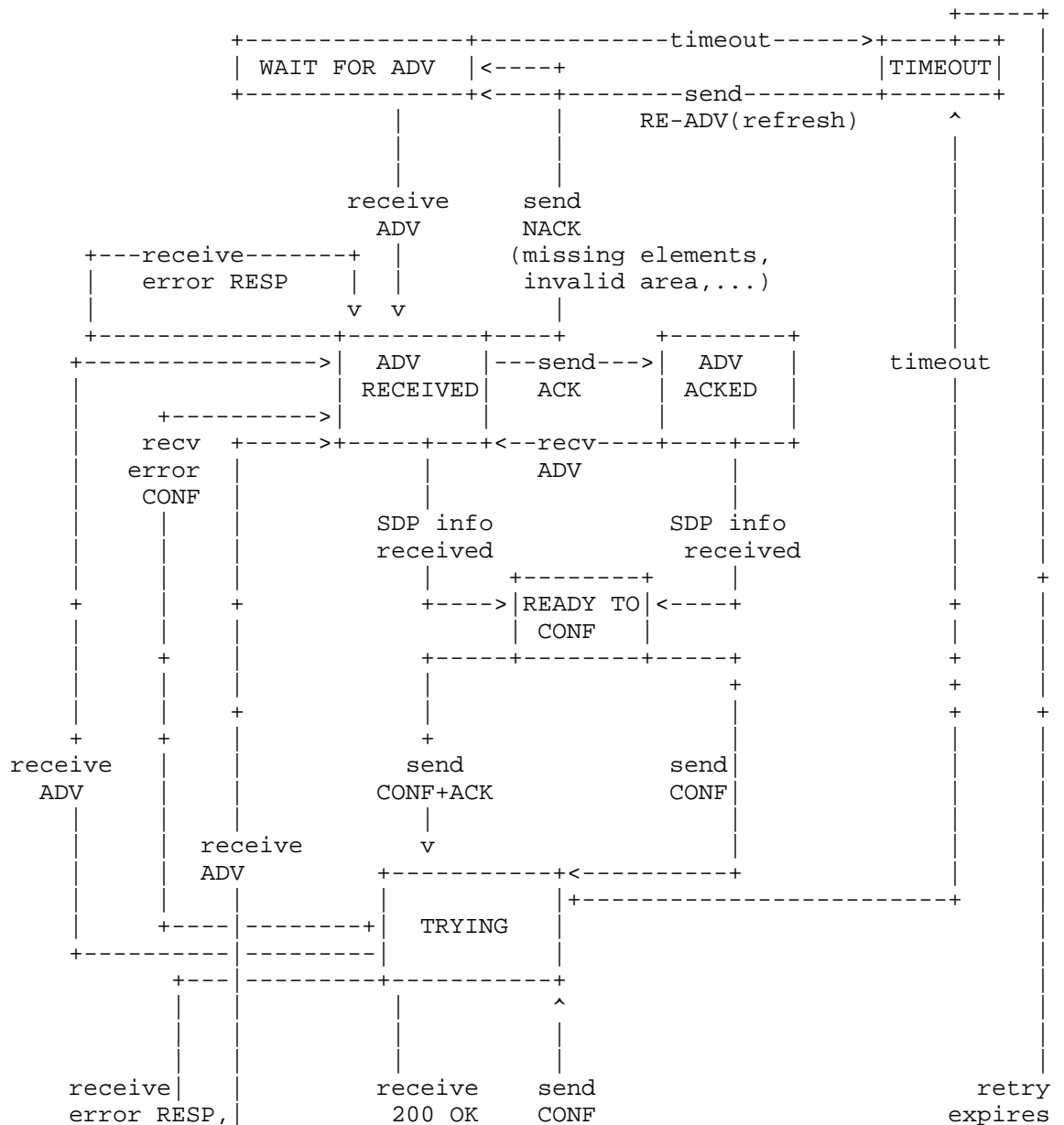
When the ADV has been received ("receive ADV"), the MC goes into the ADV RECEIVED state. The ADV is then parsed. If something goes wrong with the ADV (bad syntax, missing XML elements, etc.), the MC sends a NACK message (an ACK with an error response code) to the MP specifying the encountered problem via a proper reason phrase. In this way, the MC switches back to the WAIT FOR ADV state, waiting for a new copy of the ADV. If the ADV is successfully processed, the MC issues a successful ACK message to the MP and moves to the ADV ACKED state. When the SDP information arrives, from the ADV RECEIVED or the ADV ACKED state the MC switches to the READY TO CONF state. When the CONF request is ready, the MC sends it and moves to the TRYING state. If the ADV has not been already sent, the MC can piggyback the ACK message within the CONF request.

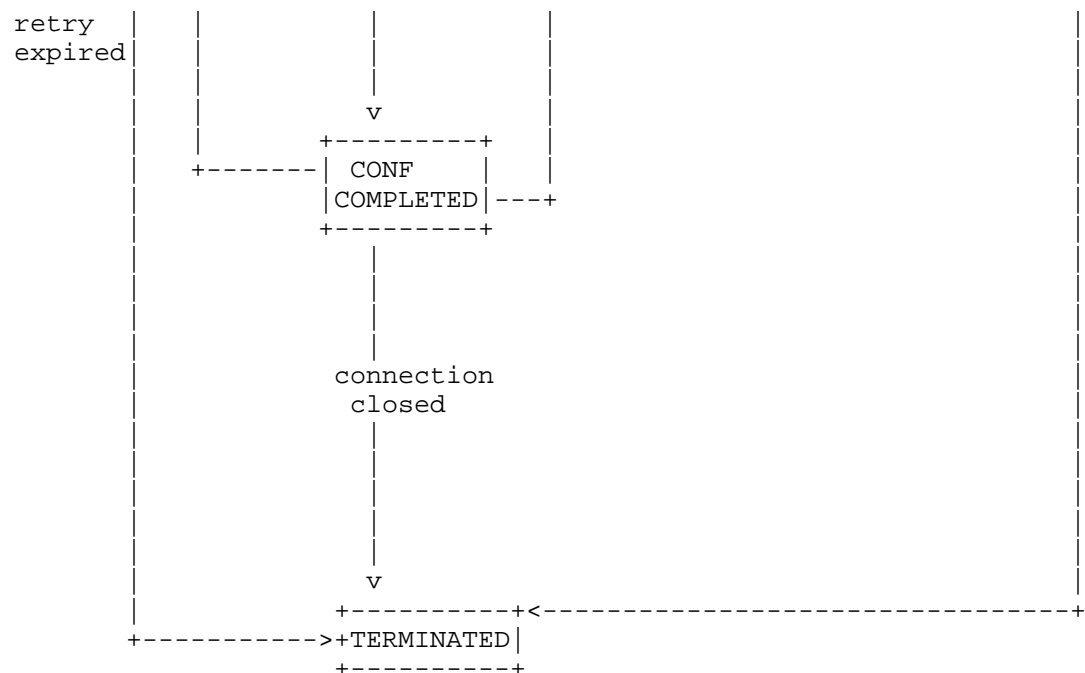
While in the TRYING state, the MC is waiting for a CONF RESPONSE message (to the issued CONF) from the MP. If the timeout expires ("timeout"), the MC moves to the TIMEOUT state and sends a READV in order to solicit a new ADV from the MP. If a CONF RESPONSE with an error code is received ("receive 4xx, 5xx not supported"), then the MC moves back to the ADV RECEIVED state and produces a new CONF message to be sent to the MP. If a successful CONF RESPONSE arrives ("receive 200 OK"), the MC gets into the CONF COMPLETED state.

When the MC is in the CONF COMPLETED state, it means that the telepresence session configuration has been set up according to the MC's preferences. Both the MP and the MC have agreed on (and are aware of) the media streams to be exchanged within the call. If the MC decides to change something in the call settings, it issues a new CONF ("send CONF") and moves back to the TRYING state. If a new ADV arrives from the MP ("receive ADV"), it means that something has changed on the MP's side. The MC then moves to the ADV RECEIVED state and prepares a new CONF taking into account the received updates. When the underlying channel is closed, the MC moves into the TERMINATED state.

The TERMINATED state is reachable from each of the aforementioned states whenever the underlying channel is closed. The corresponding

[Page 22]





6.2. Media Provider's state machine

In the PREPARING ADV state, the MP is preparing the ADV message reflecting the actual telepresence capabilities. After the ADV has been sent, the MP moves to the WAIT FOR ACK state. If the ACK arrives, the MP moves to the WAIT FOR CONF state. If a NACK arrives, it goes back to the PREPARING ADV state.

When in the WAIT FOR ACK state, if a CONF or a CONF+ACK arrives, the MP switch to the CONF RECEIVED state directly.

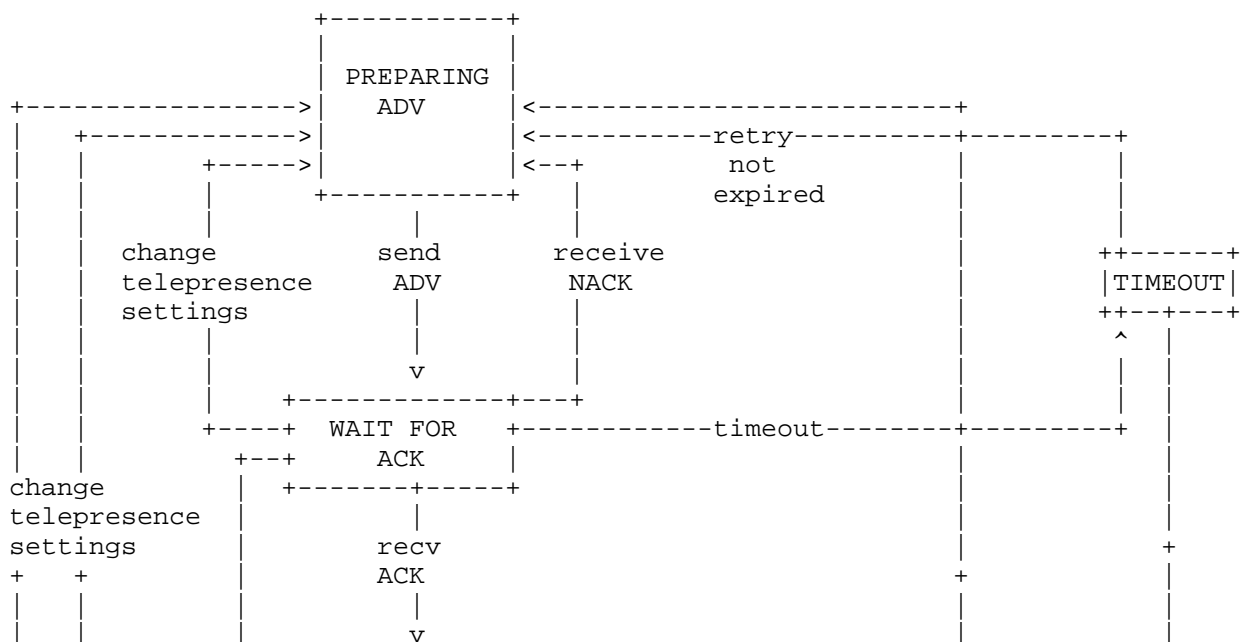
When in the WAIT FOR CONF state, the MP is listening to the channel for a CONF coming from the MC. If a RE-ADV is received, the MP goes back to the IDLE state and issues an ADV again. If telepresence settings change in the meanwhile, it moves back to the PREPARING ADV state and prepares a new ADV to be sent to the MC. If a CONF arrives, the MP switches to the CONF RECEIVED state. If nothing happens and the timeout expires, than the MC falls into the TIMEOUT state.

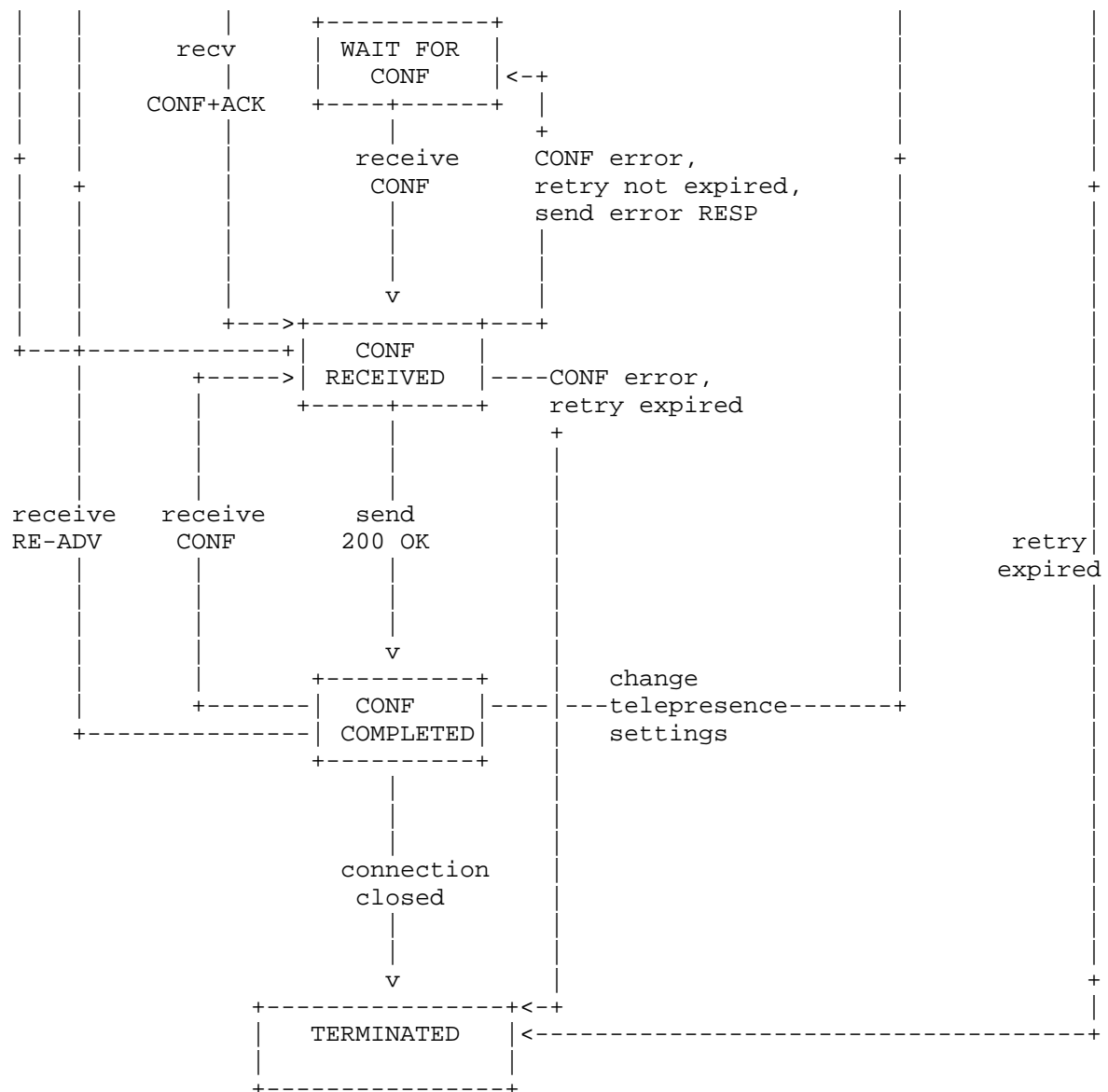
In the TIMEOUT state, if the number of trials does not exceed the retry threshold, the MC comes back to the PREPARING ADV state for sending a new ADV. Otherwise, it goes to the TERMINATED state.

The MP in the CONF RECEIVED state is processing the received CONF in order to produce a CONF RESPONSE message. If the MP is fine with the MC's configuration, then it sends back a 200 OK successful CONF RESPONSE and moves to the IN CALL state. If there are errors during CONF processing, then the MC returns a CONF RESPONSE carrying an error response code. Finally, if there are changes in the telepresence settings, it goes back to the PREPARING ADV state to issue an updated ADV.

When in the CONF COMPLETED state, the MP has successfully configured the telepresence session according to the MC's specifications. If a new CONF arrives, it switches to the CONF RECEIVED state to analyze the new request. If a RE-ADV arrives, or some modifications are applied to the telepresence options, then it moves to the PREPARE-ADV state to issue the ADV. When the channel is terminated, the MP falls into the TERMINATED state.

The TERMINATED state is reachable from each of the aforementioned states whenever the underlying channel is closed. The corresponding transitions have not been reported for the sake of simplicity. This termination condition is a temporary solution.





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 can not 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-message". 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 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 will be 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:

the information carried in the existing messages (for example, we may want to add more fields within an existing message);

the meaning of the 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:

- (i) XML elements defined within the CLUE protocol XML schema itself (protocol-specific information)
- (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:

```
elementFormDefault="qualified"
```

```
attributeFormDefault="unqualified"
```

In that case, the new information must be qualified by namespaces other than "urn:ietf:params:xml:ns:clue-message" (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 last 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 the communication parties must be aware of.

Both the types of extensions, i.e., the information and the protocol

extensions, can be characterized by:

a name;

an external XML Schema defining the XML information and/or the XML messages representing the extension;

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="xs:string" 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="0.02"
  targetNamespace="urn:ietf:params:xml:ns:clue-message"
  xmlns:tns="urn:ietf:params:xml:ns:clue-message"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:dm="urn:ietf:params:xml:ns:clue-info"
  xmlns="urn:ietf:params:xml:ns:clue-message"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <!-- Import data model schema -->
  <xs:import namespace="urn:ietf:params:xml:ns:clue-info"
```

```
schemaLocation="data-model-schema-05.xsd"/>
```

```
<!-- ELEMENT DEFINITIONS -->
```

```
<xs:element name="options" type="optionsMessageType"/>
<xs:element name="optionsResponse" type="optionsResponseMessageType"/>
<!--<xs:element name="optionsAck" type="optionsAcknowledgementMessageType"/>-->
<xs:element name="advertisement" type="advertisementMessageType"/>
<xs:element name="ack" type="advAcknowledgementMessageType"/>
<xs:element name="configure" type="configureMessageType"/>
<xs:element name="configureResponse" type="configureResponseMessageType"/>
<xs:element name="readv" type="readvMessageType"/>
<xs:element name="readvResponse" type="readvResponseMessageType"/>
```

```
<!-- 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="xs:string" use="required"/>
</xs:complexType>
```

```
<!-- 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="xs:string" 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="xs:string" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- CLUE OPTIONS RESPONSE (2 WAY) -->
<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="xs:string" 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 OPTIONS RESPONSE (3 WAYS) -->
<!-- <xs:complexType name="optionsResponseMessageType">
  <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>
-->

<!-- CLUE OPTIONS ACK (3 WAYS)-->
<!--
<xs:complexType name="optionsAckMessageType">
<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="version" type="xs:string" minOccurs="0"
    maxOccurs="1"/>
<xs:element name="commonOptions" type="supportedOptionsType" 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 fields -->
<xs:element name="mediaCaptures" type="dm:mediaCapturesType"/>
<xs:element name="encodingGroups" type="dm:encodingGroupsType"/>
<xs:element name="captureScenes" type="dm:captureScenesType"/>
<xs:element name="simultaneousSets" type="dm:simultaneousSetsType"
    minOccurs="0"/>
<xs:element name="globalCaptureEntries" type="dm:globalCaptureEntriesType"
    minOccurs="0"/>
<xs:element name="participants" type="dm:participantsType" minOccurs="0"/>
<xs:any namespace="##other" processContents="lax" minOccurs="0"/>
</xs:sequence>
<xs:anyAttribute namespace="##other" processContents="lax"/>
```

```
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- 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>

<!-- 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>

<!-- 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:integer"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
</xs:complexContent>
</xs:complexType>

<!-- CLUE READV MESSAGE TYPE -->
<xs:complexType name="readvMessageType">
  <xs:complexContent>
    <xs:extension base="clueMessageType">
      <xs:sequence>
        <xs:element name="lastReceivedAdv" type="xs:short"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- CLUE READV RESPONSE MESSAGE TYPE -->
<xs:complexType name="readvResponseMessageType">
  <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="readvSequenceNr" type="xs:string" minOccurs="0"/>
        <xs:element name="mediaCaptures" type="dm:mediaCapturesType" minOccurs="0"/>
        <xs:element name="encodingGroups" type="dm:encodingGroupsType" minOccurs="0"/>
        <xs:element name="captureScenes" type="dm:captureScenesType" minOccurs="0"/>
        <xs:element name="simultaneousSets" type="dm:simultaneousSetsType" minOccurs="0"/>
        <xs:element name="globalCaptureEntries" type="dm:globalCaptureEntriesType" minOccurs="0"/>
        <xs:element name="participants" type="dm:participantsType" minOccurs="0"/>
        <xs:any namespace="##other"
          processContents="lax" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

</xs:schema>
```

10. ADVERTISEMENT examples

In the following we provide two examples of ADVERTISEMENT representing the telepresence environment described in [I-D.ietf-clue-data-model-schema], Section 22 "Sample XML file" and Section 23 "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 22 "Sample XML file". The XML file can be downloaded here:
<http://wpage.unina.it/spromano/CLUE/>.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<advertisement xmlns="urn:ietf:params:xml:ns:clue-message"
  xmlns:ns2="urn:ietf:params:xml:ns:clue-info"
  xmlns:ns3="urn:ietf:params:xml:ns:vccard-4.0"
  protocol="CLUE" v="0.0">
  <clueId>Napoli</clueId>
  <sequenceNr>45</sequenceNr>
  <mediaCaptures>
    <ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="ns2:videoCaptureType" captureID="AC0">
      <ns2:capturedMedia>audio</ns2:capturedMedia>
      <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:maxCaptureEncodings>1</ns2:maxCaptureEncodings>
  </ns2:mediaCapture>
</ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:type="ns2:videoCaptureType" captureID="VC0">
  <ns2:capturedMedia>video</ns2:capturedMedia>
  <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:maxCaptureEncodings>2</ns2:maxCaptureEncodings>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC1">
  <ns2:capturedMedia>video</ns2:capturedMedia>
  <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:maxCaptureEncodings>2</ns2:maxCaptureEncodings>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" captureID="VC2">
  <ns2:capturedMedia>video</ns2:capturedMedia>
  <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:maxCaptureEncodings>2</ns2:maxCaptureEncodings>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" captureID="VC3">
  <ns2:capturedMedia>video</ns2:capturedMedia>
  <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:maxCaptureEncodings>1</ns2:maxCaptureEncodings>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" captureID="VC4">
  <ns2:capturedMedia>video</ns2:capturedMedia>
  <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:maxCaptureEncodings>1</ns2:maxCaptureEncodings>
</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:sceneEntries>
      <ns2:sceneEntry mediaType="video" sceneEntryID="SE1">
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC0</ns2:captureIDREF>
          <ns2:captureIDREF>VC1</ns2:captureIDREF>
          <ns2:captureIDREF>VC2</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneEntry>
      <ns2:sceneEntry mediaType="video" sceneEntryID="SE2">
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC3</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneEntry>
      <ns2:sceneEntry mediaType="video" sceneEntryID="SE3">
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC4</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneEntry>
      <ns2:sceneEntry mediaType="audio" sceneEntryID="SE4">
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC4</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneEntry>
    </ns2:sceneEntries>
  </ns2:captureScene>
</captureScenes>
<simultaneousSets>
  <ns2:simultaneousSet setID="SS1">
    <ns2:captureIDREF>VC3</ns2:captureIDREF>
    <ns2:sceneEntryIDREF>SE1</ns2:sceneEntryIDREF>
  </ns2:simultaneousSet>
  <ns2:simultaneousSet setID="SS2">
    <ns2:captureIDREF>VC0</ns2:captureIDREF>
    <ns2:captureIDREF>VC2</ns2:captureIDREF>
    <ns2:captureIDREF>VC4</ns2:captureIDREF>
    <ns2:captureIDREF>VC3</ns2:captureIDREF>
  </ns2:simultaneousSet>
</simultaneousSets>
```



```
</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>
</advertisement>
```

10.2. ADV with MCCs

The associated Media Provider's telepresence capabilities are described in [I-D.ietf-clue-data-model-schema], Section 23 "MCC example". The XML file can be downloaded here:
<http://wpage.unina.it/spromano/CLUE/>.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<advertisement xmlns="urn:ietf:params:xml:ns:clue-message"
  xmlns:ns2="urn:ietf:params:xml:ns:clue-info"
  xmlns:ns3="urn:ietf:params:xml:ns:vccard-4.0" protocol="CLUE" v="0.1">
  <clueId>Napoli CLUE Endpoint</clueId>
  <sequenceNr>34</sequenceNr>
  <mediaCaptures>
```

```
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" captureID="AC0">
  <ns2:capturedMedia>audio</ns2:capturedMedia>
  <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:maxCaptureEncodings>1</ns2:maxCaptureEncodings>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:videoCaptureType" captureID="VC0">
  <ns2:capturedMedia>video</ns2:capturedMedia>
  <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>
```

```
n>      <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:maxCaptureEncodings>2</ns2:maxCaptureEncodings>
    </ns2:mediaCapture>
    <ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC1">
      <ns2:capturedMedia>video</ns2:capturedMedia>
      <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:maxCaptureEncodings>2</ns2:maxCaptureEncodings>
    </ns2:mediaCapture>
    <ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC2">
      <ns2:capturedMedia>video</ns2:capturedMedia>
      <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:maxCaptureEncodings>2</ns2:maxCaptureEncodings>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC3">
  <ns2:capturedMedia>video</ns2:capturedMedia>
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:encGroupIDREF>EG0</ns2:encGroupIDREF>
  <ns2:nonSpatiallyDefinable>true</ns2:nonSpatiallyDefinable>
  <ns2:content>
    <ns2:sceneEntryIDREF>SE1</ns2:sceneEntryIDREF>
  </ns2:content>
  <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:maxCaptureEncodings>1</ns2:maxCaptureEncodings>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC4">
  <ns2:capturedMedia>video</ns2:capturedMedia>
  <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:maxCaptureEncodings>1</ns2:maxCaptureEncodings>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC5">
  <ns2:capturedMedia>video</ns2:capturedMedia>
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:nonSpatiallyDefinable>true</ns2:nonSpatiallyDefinable>
  <ns2:content>
    <ns2:sceneEntryIDREF>SE1</ns2:sceneEntryIDREF>
  </ns2:content>
  <ns2:composed>false</ns2:composed>
  <ns2:switched>true</ns2:switched>
  <ns2:policy>Soundlevel:1</ns2:policy>
  <ns2:maxCaptures>1</ns2:maxCaptures>
  <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:maxCaptureEncodings>1</ns2:maxCaptureEncodings>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC6">
  <ns2:capturedMedia>video</ns2:capturedMedia>
  <ns2:captureSceneIDREF>CS1</ns2:captureSceneIDREF>
  <ns2:nonSpatiallyDefinable>true</ns2:nonSpatiallyDefinable>
```

```
<ns2:content>
  <ns2:sceneEntryIDREF>SE1</ns2:sceneEntryIDREF>
</ns2:content>
<ns2:composed>false</ns2:composed>
<ns2:switched>true</ns2:switched>
<ns2:policy>Soundlevel:2</ns2:policy>
<ns2:maxCaptures>1</ns2:maxCaptures>
<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:maxCaptureEncodings>1</ns2:maxCaptureEncodings>
</ns2:mediaCapture>
<ns2:mediaCapture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:videoCaptureType" captureID="VC7">
  <ns2:capturedMedia>video</ns2:capturedMedia>
  <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: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:maxCaptureEncodings>1</ns2:maxCaptureEncodings>
</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:sceneEntries>
      <ns2:sceneEntry mediaType="video" sceneEntryID="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:sceneEntry>
      <ns2:sceneEntry mediaType="video" sceneEntryID="SE2">
        <ns2:description lang="en">loudest segment of the room
        </ns2:description>
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC3</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneEntry>
      <ns2:sceneEntry mediaType="video" sceneEntryID="SE5">
        <ns2:description lang="en">loudest segment
        of the room + pips</ns2:description>
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC7</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneEntry>
      <ns2:sceneEntry mediaType="audio" sceneEntryID="SE4">
        <ns2:description lang="en">room audio</ns2:description>
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>AC0</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneEntry>
      <ns2:sceneEntry mediaType="video" sceneEntryID="SE3">
        <ns2:description lang="en">room video</ns2:description>
        <ns2:mediaCaptureIDs>
          <ns2:captureIDREF>VC4</ns2:captureIDREF>
        </ns2:mediaCaptureIDs>
      </ns2:sceneEntry>
    </ns2:sceneEntries>
  </ns2:captureScene>
</captureScenes>
<simultaneousSets>
```

```
<ns2:simultaneousSet setID="SS1">
  <ns2:captureIDREF>VC7</ns2:captureIDREF>
  <ns2:sceneEntryIDREF>SE1</ns2:sceneEntryIDREF>
</ns2:simultaneousSet>
<ns2:simultaneousSet setID="SS2">
  <ns2:captureIDREF>VC0</ns2:captureIDREF>
  <ns2:captureIDREF>VC2</ns2:captureIDREF>
  <ns2:captureIDREF>VC4</ns2:captureIDREF>
  <ns2:captureIDREF>VC7</ns2:captureIDREF>
</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>
</advertisement>
```

11. 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.
12. Diff with draft-presta-clue-protocol-04
1. The response code type error in the OPTIONS response (and in other parts) has been corrected.
13. 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.kyzivat-clue-signaling].
 4. Besides the two way one, a three way mechanism for the options negotiation has been proposed and provided to foster discussion.
14. 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.
15. Acknowledgments
- The authors thank all the CLUErs for their precious feedbacks and support, in particular Paul Kyzivat, Christian Groves and Scarlett Liuyan.

16. Informative 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-05 (work in progress), June 2014.
- [I-D.ietf-clue-datachannel] Holmberg, C., "CLUE Protocol Data Channel", draft-ietf-clue-datachannel-00 (work in progress), March 2014.
- [I-D.ietf-clue-framework] Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-15 (work in progress), May 2014.
- [I-D.ietf-clue-telepresence-requirements] Romanow, A., Botzko, S., and M. Barnes, "Requirements for Telepresence Multi-Streams", draft-ietf-clue-telepresence-requirements-07 (work in progress), December 2013.
- [I-D.kyzivat-clue-signaling] Kyzivat, P., Xiao, L., Groves, C., and R. Hansen, "CLUE Signaling", draft-kyzivat-clue-signaling-08 (work in progress), April 2014.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [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.

[RFC5261]

Urpalainen, J., "An
Extensible Markup Language
(XML) Patch Operations
Framework Utilizing XML
Path Language (XPath)
Selectors", RFC 5261,
September 2008.

[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.

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

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

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

CLUE Signaling
draft-ietf-clue-signaling-02

Abstract

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

Status of this Memo

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

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

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

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

Copyright Notice

Copyright (c) 2014 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. Media Feature Tag Definition	5
4. SDP Grouping Framework CLUE Extension Semantics	5
4.1. General	5
4.2. The CLUE data channel and the CLUE grouping semantic	6
4.3. CLUE-controlled media and the CLUE grouping semantic	6
4.4. SDP semantics for CLUE-controlled media	6
4.4.1. Signalling CLUE Encodings	7
4.4.1.1. Referencing encodings in the CLUE protocol	7
4.4.1.2. Media line directionality	8
4.4.2. Negotiating receipt of CLUE capture encodings in SDP	8
4.5. SDP Offer/Answer Procedures	9
4.5.1. Generating the Initial Offer	9
4.5.2. Generating the Answer	9
4.5.2.1. Negotiating use of CLUE and the CLUE data channel	9
4.5.2.2. Negotiating CLUE-controlled media	9
4.5.2.3. Negotiating non-CLUE controlled media	10
4.5.3. Processing the initial Offer/Answer negotiation	10
4.5.3.1. Successful CLUE negotiation	10
4.5.3.2. CLUE negotiation failure	11
4.5.4. Modifying the session	11
4.5.4.1. Adding and removing CLUE-controlled media	11
4.5.4.2. Enabling CLUE mid-call	11
4.5.4.3. Disabling CLUE mid-call	12
5. Interaction of CLUE protocol and SDP negotiations	12
5.1. Independence of SDP and CLUE negotiation	12
5.2. Constraints on sending media	13
5.3. Recommendations for operating with non-atomic operations	13
6. Multiplexing of CLUE-controlled media using BUNDLE	14
6.1. Overview	14
6.2. Usage of BUNDLE with CLUE	15
6.2.1. Generating the Initial Offer	15
6.2.2. Bundle Address Synchronization	15
6.2.3. Multiplexing of the data channel and RTP media	15
7. Example: A call between two CLUE-capable endpoints	15
8. Example: A call between a CLUE-capable and non-CLUE endpoint	23

9. CLUE requirements on SDP O/A	25
10. SIP Signaling	25
11. CLUE over RTCWEB	25
12. Open Issues	25
13. What else?	26
14. Acknowledgements	26
15. IANA Considerations	26
16. Security Considerations	26
17. Change History	26
18. References	29
18.1. Normative References	29
18.2. Informative References	30
Authors' Addresses	31

1. Introduction

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

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

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

2. Terminology

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

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

Other terms introduced here:

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

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

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

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

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

3. Media Feature Tag Definition

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

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

4. SDP Grouping Framework CLUE Extension Semantics

4.1. General

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

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

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

4.2. The CLUE data channel and the CLUE grouping semantic

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

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

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

4.3. CLUE-controlled media and the CLUE grouping semantic

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

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

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

4.4. SDP semantics for CLUE-controlled media

4.4.1. Signalling CLUE Encodings

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

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

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

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

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

4.4.1.1. Referencing encodings in the CLUE protocol

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

Each <encID> element in a CLUE ADVERTISEMENT message SHOULD 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 <encID> element in a CLUE CONFIGURE message SHOULD

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

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

4.4.1.2. Media line directionality

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

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

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

4.5. SDP Offer/Answer Procedures

4.5.1. Generating the Initial Offer

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

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

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

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

4.5.2. Generating the Answer

4.5.2.1. Negotiating use of CLUE and the CLUE data channel

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

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

4.5.2.2. Negotiating CLUE-controlled media

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

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

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

4.5.2.3. 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.

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

4.5.3. Processing the initial Offer/Answer negotiation

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

4.5.3.1. Successful CLUE negotiation

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

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

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

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

4.5.3.2. CLUE negotiation failure

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

4.5.4. Modifying the session

4.5.4.1. Adding and removing CLUE-controlled media

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

Subsequent offer/answer exchanges MAY also 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 CLUE-controlled media that has been successfully negotiated. Implementations can decide if they wish to disable non-CLUE controlled media once the call has been made CLUE enabled, or to wait until sending of the CLUE-controlled media has been successfully negotiated.

4.5.4.2. Enabling CLUE mid-call

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

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

4.5.4.3. Disabling CLUE mid-call

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

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

5. Interaction of CLUE protocol and SDP negotiations

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

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

5.1. Independence of SDP and CLUE negotiation

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

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

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

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

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

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

5.2. Constraints on sending media

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

5.3. Recommendations for operating with non-atomic operations

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

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

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

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

6. Multiplexing of CLUE-controlled media using BUNDLE

6.1. Overview

A CLUE call may involve sending and/or receiving significant numbers of media streams. Conventionally, media streams are sent and received on unique ports. However, each 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.

6.2. Usage of BUNDLE with CLUE

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

6.2.1. Generating the Initial Offer

BUNDLE mandates that the initial SDP offer MUST use a unique address for each m-line with a non-zero port. Because CLUE implementations 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.

6.2.2. Bundle Address Synchronization

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

6.2.3. Multiplexing of the data channel and RTP media

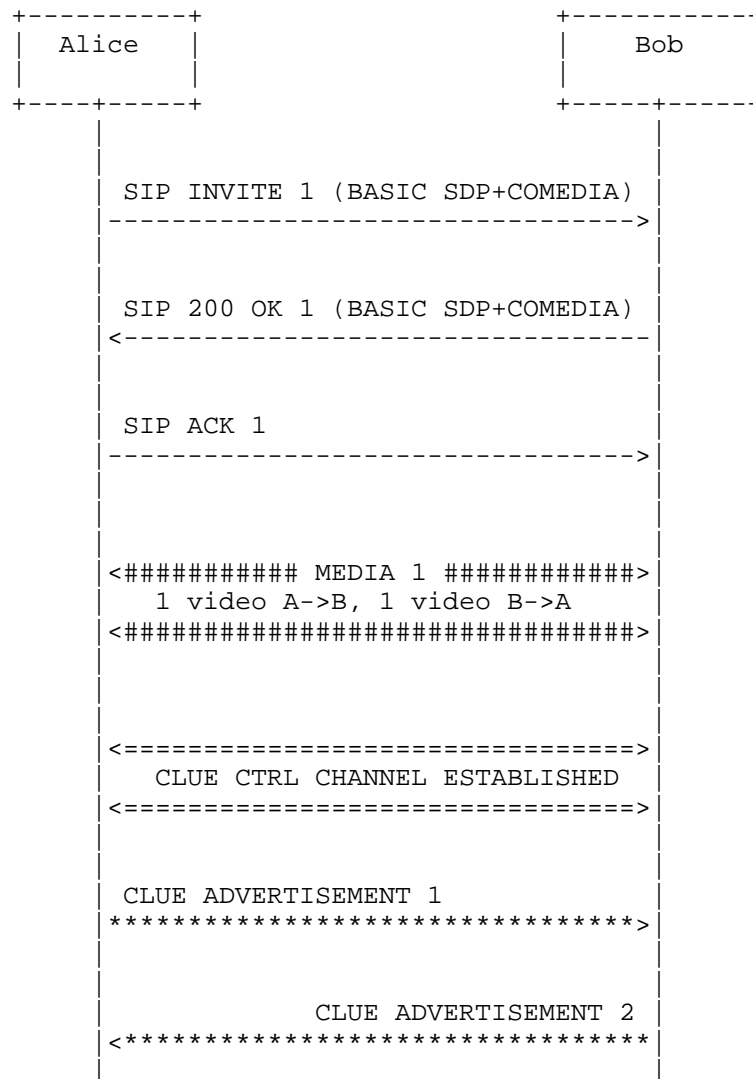
BUNDLE-supporting CLUE-capable devices MAY include the data channel in the same BUNDLE group as RTP media. In this case the device MUST be able to demultiplex the various transports - see section 7.2 of the BUNDLE draft [I-D.ietf-mmusic-sdp-bundle-negotiation]. If the BUNDLE group includes other protocols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

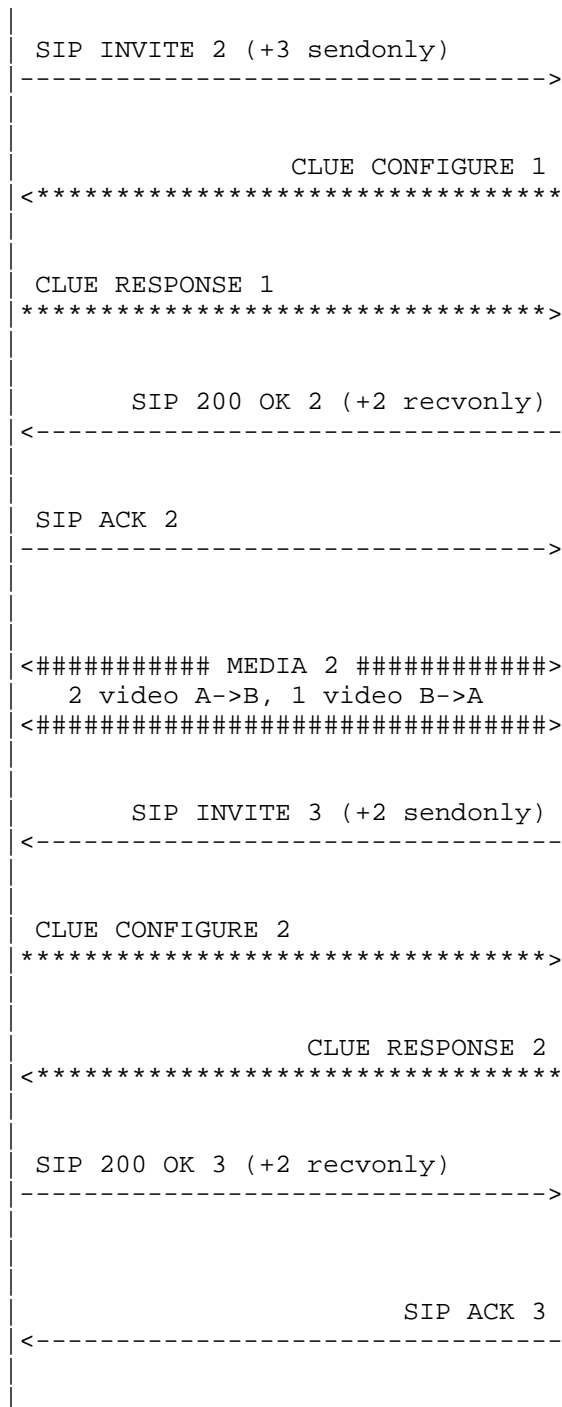
7. Example: A call between two CLUE-capable endpoints

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

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

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





```

|<##### MEDIA 3 #####>|
| 2 video A->B, 2 video B->A |
|<#####>|
|
|
|
v
v

```

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

```

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

```

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

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

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

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

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

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

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


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

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

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

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

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

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

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

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

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

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

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

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

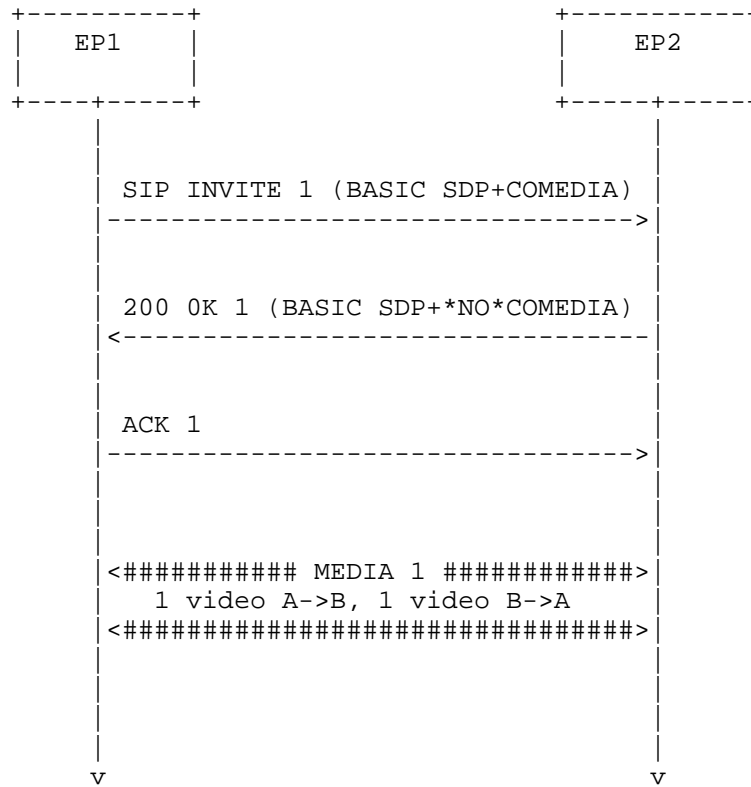
Finally, on receiving 200 OK 3 Bob is now able to send the two streams of video Alice requested - this is illustrated as MEDIA 3.

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

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

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

protocol.



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

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

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

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

9. CLUE requirements on SDP O/A

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

10. SIP Signaling

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

11. CLUE over RTCWEB

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

12. Open Issues

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

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

13. What else?

14. Acknowledgements

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

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

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

15. IANA Considerations

TBD

16. Security Considerations

TBD

17. Change History

-02: Revision by Rob Hansen

- * Added section on not accepting non-CLUE-controlled "m" lines in the initial answer when CLUE is to be negotiated.
- * Removed previous language attempting to describe media restrictions for CLUE-controlled "m" lines that had not been configured, and replaced it with much more accurate 'treat as "a=inactive" was set'.
- * Made label element mandatory for CLUE-controlled media (was previously "SHOULD include", but there didn't seem a good reason for this - anyone wishing to include the "m" line but not immediately use it in CLUE can simply leave it out of the <encodingIDList>.)
- * Added a section on the specifics of relating encodings in SDP to <encID> elements in the CLUE protocol, including the fact that both ADVERTISEMENTS 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.presta-clue-protocol], but that has not yet been resolved in those documents, into an appendix.
- draft-kyzivat-06: Revisions by Rob Hansen
- * Removed CLUE message XML schema and details that are now in draft-presta-clue-protocol
 - * Encoding limits in SDP section updated to note that this has been investigated and discussed and is the current working assumption of the WG, though consensus has not been fully achieved.
 - * A section has also been added on the current mandation of unidirectional "m"-lines.
 - * Updated CLUE messaging in example callflow to match draft-presta-clue-protocol-03
- draft-kyzivat-05: Revisions by pkyzivat:
- * Specified versioning model and mechanism.
 - * Added explicit response to all messages.
 - * Rearranged text to work with the above changes. (Which rendered diff almost useless.)
- draft-kyzivat-04: Revisions by Rob Hansen: ???
- draft-kyzivat-03: Revisions by pkyzivat:
- * Added a syntax section with an XML schema for CLUE messages. This is a strawhorse, and is very incomplete, but it establishes a template for doing this based on elements defined in the data model. (Thanks to Roberta for help with this!)
 - * Did some rewording to fit the syntax section in and reference it.
 - * Did some relatively minor restructuring of the document to make it flow better in a logical way.
- draft-kyzivat-02: A bunch of revisions by pkyzivat:
- * Moved roberta's call flows to a more appropriate place in the document.
 - * New section on versioning.
 - * New section on NAK.
 - * A couple of possible alternatives for message acknowledgment.
 - * Some discussion of when/how to signal changes in provider state.
 - * Some discussion about the handling of transport errors.
 - * Added a change history section.
- These were developed by Lennard Xiao, Christian Groves and Paul, so added Lennard and Christian as authors.

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

18. References

18.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [I-D.ietf-clue-framework]
Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-16 (work in progress), June 2014.
- [I-D.presta-clue-data-model-schema]
Presta, R. and S. Romano, "An XML Schema for the CLUE data model", draft-presta-clue-data-model-schema-03 (work in progress), March 2013.
- [I-D.presta-clue-protocol]
Presta, R. and S. Romano, "CLUE protocol", draft-presta-clue-protocol-04 (work in progress), May 2014.
- [I-D.ietf-clue-datachannel]
Holmberg, C., "CLUE Protocol Data Channel", draft-ietf-clue-datachannel-00 (work in progress), March 2014.
- [I-D.groves-clue-latent-config]
Groves, C., Yang, W., and R. Even, "CLUE and latent configurations", draft-groves-clue-latent-config-00 (work in progress), January 2014.
- [I-D.ietf-mmusic-sctp-sdp]
Loreto, S. and G. Camarillo, "Stream Control Transmission Protocol (SCTP)-Based Media Transport in the Session Description Protocol (SDP)", draft-ietf-mmusic-sctp-sdp-06 (work in progress), February 2014.
- [I-D.tuexen-tsvwg-sctp-dtls-encaps]
Jesup, R., Loreto, S., Stewart, R., and M. Tuexen, "DTLS

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

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

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

18.2. Informative References

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

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

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

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

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

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

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

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

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

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

Authors' Addresses

Paul Kyzivat
Huawei

Email: pkyzivat@alum.mit.edu

Lennard Xiao
Huawei

Email: lennard.xiao@huawei.com

Christian Groves
Huawei

Email: Christian.Groves@nteczone.com

Robert Hansen
Cisco Systems

Email: rohanse2@cisco.com

